Harper Adams University

FIELD ROBOT EVENT

PROCEEDINGS 2017

15th Ed.

Proceedings of the 15th Field Robot Event 13th June – 16th June 2017

Harper Adams University
Newport
Shropshire
TF10 8NB
United Kingdom

Department of Engineering

Published February 2018

Organising Team

Event Organisers: Professor Simon Blackmore, Dr Ianto Guy, Rose Steele and Sam Wane - Harper Adams University

The Organising Team:

Timo Oksanen – University of Helsinki

Han-Werner Griepentrog – University of Hohenheim

Eldert van Henten – Wageningen University

Arno Ruckelsjausen – Osnabruk University

The Challenge Organising Team (UK): Ianto Guy; Simon Blackmore; Parmjit Chima; Sam Wane; Tom Underhill; Kit Franklin; Dave Coleman; Michael Warbrick

Admin support: Rose Steele and Faye Vale

Student Support: Jack Clements and Yasmin Wulkau

Editor Programme & Proceedings: Christopher Waghorn

Contents

Event Introduction

The Field Robotic Event is an annual competition, launched in 2003 by the University of Wageningen in the Netherlands. Harper Adams University hosted the 15th Annual Event. The competition philosophy is to boost innovation of future technologies for Precision Farming under 'real conditions' in the field.

The competition requires teams to design, construct and demonstrate an autonomous robot to complete a variety of agricultural tasks autonomously. Tasks are challenging, demonstrating sensing, navigation and actuation. Each task is scored and ranked.

A total of 14 teams from all over the world participated in this year's event.

These proceedings outline the requirements of each task, the results of the competition and a summary of the technology used in each team's robot. Where a team has submitted their own technical report for their robot, this has been incorporated unabridged with a small amount of formatting into this document.

Event Tasks.

Comprehensive competitions rules and regulations can be downloaded from http://www.harper-adams.ac.uk/events/fre/files/rules-and-regulations.pdf

Task 1 - "Basic navigation"

General description

For this task the robot has three minutes to navigate as far as possible between the rows of maize plants, starting in the first row and travelling sequentially into rows 2, 3, 4 etc. (figure 2). On the headland, the robot has to turn within the 2m field boundary and return in the adjacent row. This task is all about accuracy, smoothness and speed of the navigation operation between the rows.

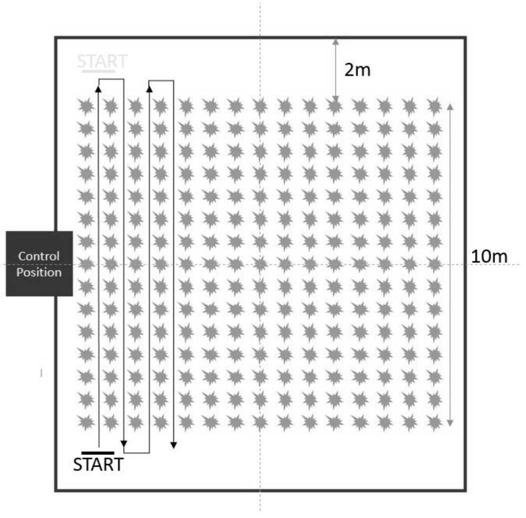


Figure 1: Task 1 Basic Navigation

Field conditions

Random stones will be placed along the path to represent a realistic field scenario. The stones will not exceed 25 mm from the average ground level. The stones may be small pebbles (diameter <25 mm) laid in the ground and large rocks that push (max 25 mm) out from the ground, both are installed. In other words, the robot must have ground clearance of this amplitude at minimum, and the robot must be able to climb over obstacles of max 25 mm height.

Rules for robots

The robot will start the task from the start line. The start line may be on the left or right of the field. The position of the start line will be notified to the teams before the start of the task.

If the robot is about to deviate out from the path and hit maize plants, the team member with the remote controller must press STOP button immediately. The STOP button must be pressed before the robot damages stems of the maize plants.

Penalties

Crop plant damage by the robot will result in a penalty of 1 meter per plant.
Manual intervention to move or adjust the robot will result in a penalty of 1 meter for each time the robot is STOPPED.

Assessment

The distance travelled in 3 minutes is measured. If the end of the field is reached in less than 3 minutes the remaining time will be used to calculate a bonus factor = total distance x 3minutes/measured time.

The total distance includes travelled distance and the penalty values. Distance and time are measured by the jury officials.

The task completing teams will be ranked by according to the total distance values.

The best 3 teams will be rewarded.

Task 2 - “Advanced navigation”

General description

Under real field conditions crop plant growth is not uniform. Sometimes plants may fail to germinate or may be attacked by pests. We will approach these field conditions in the second task.

As in task 1 the aim is to navigate as far as possible between the rows within 3 minutes.However in this task the robots have to follow a certain predefined path across the field. Additionally at some locations, plants will be missing (gaps) at either one or both sides with a maximum length of 1 meter. There will be no gaps at row entries. The robot must drive the paths in the order given before the start of the task. The code of the path pattern through the maize field is done as follows: S means START, L means LEFT hand turn, R means RIGHT hand turn and F means FINISH. The number before the L or R represents the row that has to be entered after the turn. Therefore, 2L means: Enter the second row after a left hand turn, 3R means: Enter the third row after a right hand turn. The code for a path pattern for example may be given as: S - 3L - 2L - 2R - 1R - 5L - F.

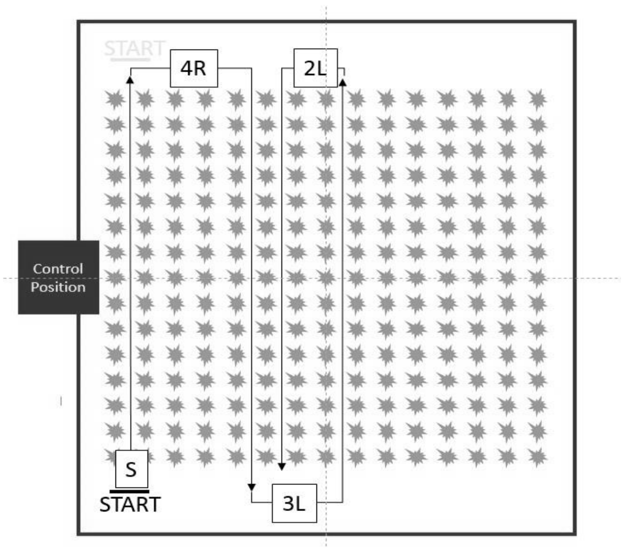


Figure 2: Task 2 Advanced Navigation

The code of the path pattern is made available to the competitors 15 minutes before putting all robots into the parc fermé. Therefore, the teams will not get the opportunity to test it in the contest field.

Field conditions

Random stones are placed along the path, to represent realistic field scenario where the robot should cope with holes etc. The stones are not exceeding the level of 35 mm from the average ground level in the neighbourhood. The stones may be pebbles (diameter <35mm) laid in the ground and large rocks that push (max 35 mm) out from the ground, both are installed. In other words, the robot must have ground clearance of this amplitude at minimum, and the robot must be able to climb over obstacles of max 35mm high. No maize plants are intentionally missing in the end of the rows. However, due to circumstances of previous runs by other robots, it is possible that some plants in the end of the rows are damaged.

Penalties

Crop plant damage by the robot will result in a penalty of 1 meter per plant. Manual intervention to move or adjust the robot will result in a penalty of 1 meter for each time the robot is STOPPED. The robot must be STOPPED if it navigates into the wrong row.

Assessment

The distance travelled in 3 minutes is measured. If the end of the field is reached in less time, this time will be used to calculate a bonus factor = total distance x 3minutes/measured time.

The total distance includes travelled distance and the penalty values. Distance and time will be measured by the jury officials.

The task completing teams will be ranked by according to the total distance values. The best 3 teams will be rewarded.

Task 3 - "Field mapping"

General description

In this task teams have 5 minutes to map the field using autonomous systems, recording the positions of weeds represented by pink golf balls and obstacles represented by yellow tennis balls. Task 3 is conducted on the area used in tasks 1 and 2. The map created in this task will be used in task 4. Up to ten obstacles may be placed in the field, either between rows or in the headland. Obstacles must not be passed regardless of whether the robot can do so without touching them. Up to ten weeds may be placed in the field. All weeds will be placed between rows.

Field conditions: As in task 2 random stones are placed along the path, to represent realistic field scenario where the robot should cope with holes etc. The stones are not exceeding the level of 35 mm from the average ground level in the neighbourhood. The stones may be pebbles (diameter <35mm) laid in the ground and large rocks that push (max 35 mm) out from the ground, both are installed. In other words, the robot must have ground clearance of this amplitude at minimum, and the robot must be able to climb over obstacles of max 35mm high. No maize plants are intentionally missing in the end of the rows. However, due to circumstances of previous runs by other robots, it is possible that some plants in the end of the rows are damaged.

The weeds are objects represented by pink golf balls randomly distributed between the rows in the soil so that only the upper half is visible. Robots may drive across or over them without a penalty. The weeds are located in a band 60 cm wide between the rows. No weeds are located within rows or on headlands. A possible example is illustrated in figure 4.

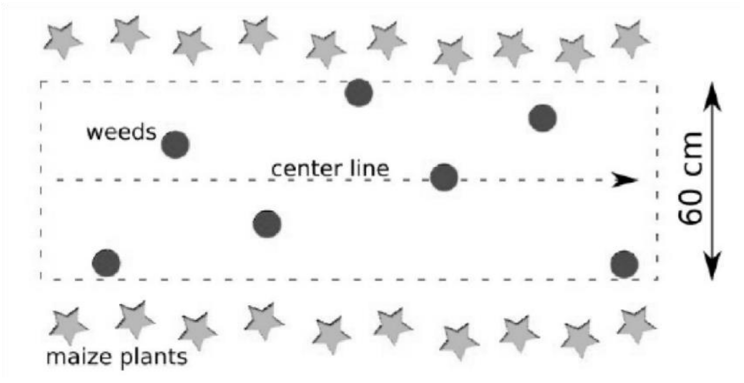


Figure 3: Possible locations of weeds for tasks 3 and 4

Obstacles are represented by yellow tennis balls which will be placed randomly between rows and on the headland. Robots are not permitted to touch or pass the obstacles.

Rules for robots

For this task teams are permitted to use systems other than the main robot, for example an unmanned aerial vehicle or a swarm of small robots. Any system used in this task must still operate autonomously.

Each team has only one attempt. The maximum available time for the run is 5 minutes.

Points will be awarded for detecting weeds and obstacles and for recording their positions.

Teams can nominate whether they wish to indicate the detection of weeds and obstacles separately from the mapping of their locations. Once the nomination has been made then that method must be used for the task.

There is no requirement for the robot to travel along every row, provided that all obstacles and weeds are detected, i.e. it is acceptable for example to have a robot with a high mounted camera which is capable of surveying two or three rows at a time.

Option 1

A single robot navigates between the rows, as in tasks 1 and 2, giving an audible signal when it comes across each weed or obstacle to indicate that it has detected it at that location. The detection of a weed should be indicated by a two second signal and the detection of an obstacle should be indicated by a five second signal. A robot that is capable of surveying more than one row at a time must indicate the row in which it has detected the obstacle or weed.

A robot producing an acoustic signal without any reason will be regarded as a false positive. Failure to produce an acoustic signal when an obstacle or weed is encountered will be regarded as a false negative.

The robot should have some means of storing the locations of the weeds and obstacles as this information will be required to complete task 4.

Option 2

A single robot, a swarm of robots or an unmanned aerial vehicle survey the field to produce a map which indicates the positions of weeds and obstacles in graphical form. The same rules for false positives and negatives will be applied as in option1. The Jury will judge whether the positions of the weeds and obstacles shown on the map are accurate. The map may be generated in real time or can be shown to the Jury at the end of the run. The team will have 2 minutes from the end of the run to produce the map and show it to the Jury.

The map will be required to complete task 4.

Penalties: Crop plant damage by the robot will result in a penalty of 2 points per plant.

Manual intervention to move or adjust the robot will result in a penalty of 2 points for each time the robot is STOPPED.
Indicating the presence of a weed or obstacle when none is present in that location (false positives) will result in a penalty of 1 point per occurrence.
Failure to indicate the presence of a weed or obstacle when one is present (false negatives) will result in a penalty of 2 points per occurrence.

Assessment: The Jury will register the number of true positives, false positives and false negatives:
Each correctly identified and located weed or obstacle (true positives) will be awarded 6 points per weed or obstacle.

The total travelled distance will not be assessed.

If a team completes the task in less than 5 minutes (excluding the 2 minutes allowed to produce a map), this time will be used to calculate a bonus factor = total points x 5minutes/measured time. The task completing teams will be ranked by the number of points as described above.

The three best teams will be rewarded.

Task 4 - “Weeding”

General description: In this task the main robot should be equipped with a crop sprayer capable of spraying water. The robot will use the map created in task 3 to produce an optimised path that allows it to spray all of the weeds in the shortest possible time. Teams will be allowed 10 minutes to configure their robot for spraying and load an optimised path into its navigation system. The path optimisation process can be completed using a computer which is independent of the main robot, but this process must be completed within the 10 minute time window.

The robots shall precisely spray the weeds mapped in task 3. It is not permitted to touch or pass the yellow tennis balls.

Field conditions: As in task 2 and 3 random stones are placed along the path, to represent realistic field scenario where the robot should cope with holes etc. The stones are not exceeding the level of 35 mm from the average ground level in the neighbourhood. The stones may be pebbles (diameter <35mm) laid in the ground and large rocks that push (max 35 mm) out from the ground, both are installed. In other words, the robot must have ground clearance of this amplitude at minimum, and the robot must be able to climb over obstacles of max 35mm high. No maize plants are intentionally missing in the end of the rows. However, due to circumstances of previous runs by other robots, it is possible that some plants in the end of the rows are damaged.

The weeds are objects represented by pink golf balls randomly distributed between the rows in the soil that only the upper half is visible. Robots may drive across or over them without a penalty. The weeds are located in a centred band of 60 cm width between the rows. No weeds are located within rows and on headlands.

Obstacles are represented by yellow tennis balls which will be placed randomly between rows and on the headland. Robots are not permitted to touch or pass the obstacles. The location of the obstacles and weeds will be the same in tasks 3 and 4.

As in task 3, there is no requirement for the robot to drive along every row, provided all weeds are sprayed.

Rules for robots

Each robot has only one attempt. The maximum available time for the run is 3 minutes. The robot must give an audible signal when the sprayer is operated.

The robot must spray only the weeds or the circular area around the golf ball with a diameter of 25 cm. Spraying outside this weed circle is counted as false positive, with no true positive scoring. In the case that the robot is spraying or producing an acoustic signal without any reason, this is regarded as false positive.

Penalties: Crop plant damage by the robot will result in a penalty of 2 points per plant.

Manual intervention to move or adjust the robot will result in a penalty of 2 points for each time the robot is STOPPED.

Activating the sprayer or making an audible signal when no weed is present in that location (false positives) will result in a penalty of 1 point per occurrence.

Failure to spray a weed when one is present (false negatives) will result in a penalty of 2 points per occurrence.

Assessment: The Jury will register the number of true positives, false positives and false negatives:

Each time a weed is sprayed correctly with the appropriate audible signal (true positives) 6 points will be awarded. If a weed is sprayed correctly but without an audible signal 4 points will be awarded.The total travelled distance will not be assessed.

If a team completes the task in less than 3 minutes, this time will be used to calculate a bonus factor = total points x 3minutes/measured time.

The task completing teams will be ranked by the number of points as described above.The three best teams will be rewarded.

Task 5 - “Freestyle”

Description

Teams are invited to let their robots perform a freestyle operation. Creativity and fun is required for this task as well as an application-oriented performance. One team member has to present the idea, the realization and perhaps to comment the robot’s performance to the jury and the audience. The freestyle task should be related to an agricultural application. Teams will have a time limit of 10 minutes for the presentation including the robot’s performance.

Assessment: The jury will assess the (i) agronomic idea, the (ii) technical complexity and the (iii) robot performance by giving points from 0 (insufficient) to 10 (excellent) for each.

The total points will be calculated using the following formula: (agronomic idea + technical complexity) x performance.

Task 5 is optional and will be awarded separately. It will not contribute to the overall competition results.

Competition Results

Summary

Task	1st Place	2nd Place	3rd Place
1. Basic Navigation	Floribot 2017	TAFR Team	Team Unicorn
2. Advanced Navigation	Team Unicorn	Kamaro Engineering	Cornholio
3. Field Mapping	Kamaro Engineering	TAFR Team	Carbonite
4. Weeding	*Joint 1st* Zukbot, Carbonite	Robatic Group	NA
5. Freestyle	Kamaro Engineering	TAFR Team	*Joint 3rd* Team Unicorn, Cornholio
Overall Winners	Kamaro Engineering	Carbonite	Floribot 2017

FRE 2017 Overall Team Ranking

Team	Task 1 Points	Task 2 Points	Task 3 Points	Task 4 Points	Total Points	Overall Rank	Task 5 Freestyle Rank
TAFR Team	**28**	**19**	**28**	**22**	**97**	**4**	2
Floribot 2017	**30**	**24**	**21**	**24**	**99**	**3**	
Cornholio	**20**	**26**	**23**	**24**	**93**	**5**	3
UniCorn	**26**	**30**	**0**	**0**	**56**	**11**	3
Agrifac Bullseye	**17**	**21**	**26**	**26**	**90**	**7**	
Dora the explorer	**16**	**0**	**22**	**0**	**38**	**13**	
Zukbot	**19**	**20**	**24**	**30**	**93**	**5**	
Voltan	**18**	**23**	**18**	**0**	**59**	**10**	
Beteigeuze - Kamaro Eng.	**23**	**28**	**30**	**25**	**106**	**1**	1
Terra	**24**	**18**	**0**	**0**	**42**	**12**	
Hefty	**21**	**25**	**20**	**0**	**66**	**9**	
Carbonite	**22**	**23**	**26**	**30**	**101**	**2**	
Helios	**25**	**17**	**19**	**21**	**82**	**8**	
K9	**0**	**0**	**0**	**0**	**0**	**14**	

The Competing Robots

Agrifac Bullseye

Name of Institution	Wageningen University & Research
Department	Farm Technology Group (FTE)
Country	The Netherlands
City	Wageningen
Webpage	www.wur.nl<http://www.wur.nl www.robatic.nl<http://www.robatic.nl
Team members	Tijn van den Aker, Rick van Essen, André Hulsman, Ton Kaarsgaren (C), Dirk Otten and Carsten Schep
Instructors	ing. SK (Sam) Blaauw, Roel Dohmen and dr. ir. JMM (Joris) IJsselmuiden
Contact Email	robatic.bullseye@gmail.com

Weight (kg)	35
Actuators/motors installed	4x Maxon Motor RE 25 and 4x Maxon Motor RE40 controlled by 4x Motion Mind Rev 2 and 4x Maxon Motor EPOS2
Turning radius (cm)	0
Battery voltage	22.2V
Battery duration (mins)	60
(W x L x H) (cm)	40 x 110 x 53

Robot software description
The software runs in Robot Operating System (ROS) Kinetic on Ubuntu 16.04LTS and is programmed in Python.
Robot hardware description
The 2D Sick LMS-111 laser scanner data is processed by an MSI Z871 motherboard running an Intel Core i7-4770T processor. To control the 4 independ?ently driven and steered wheels, 4 Maxon Motors are used EPOS2 controllers are used for propulsion and 4 Motion Mind Rev2 controllers for steering. Two additional Arduinos are used to operate the emergency stop and the spraying boom. The whole system is powered by two 22.2V 6S 20C 5000mAh Li-Po batteries.
Task strategy description
The Maize rows are scanned with the laser scanner facing forward, with these measurements the desired motion is calculated. When headland is reached, the laser scanner on the rear of the robot takes over, it detects the next rows and navigates through one of them. This way, the robot doesn't have to turn 180 degrees at the headland and our four-wheel steered platform is used optimally. Weeds and obstacles are detected with four RGB cameras using the HSV colour space.

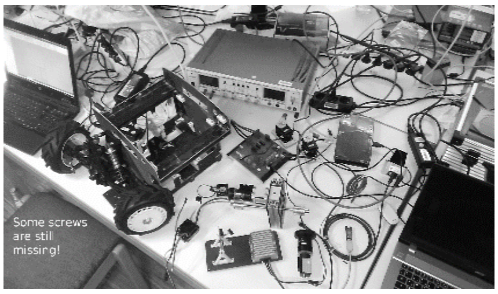
Beteigeuze

Name of Institution	Kamaro Engineering e.V. at Karlsruhe Institute of Technology
Department	MOBIMA/ FAST
Country	Germany
City	Karlsruhe
Webpage	https://kamaro-engineering.de/
Team members	Carsten Naber, Anton Schirg, Lennart Nachtigall, Michael Fuerst (C), Carlos Garcia, Friedolin Groeger, Thomas Friedel, Manuel Muth, Lukas Glueck, Christian Kinzig, Sebastian Blickle, Erik Wustmann, Marco Buckpesch, Annabell Gessinger, Christoph Breuner
Instructors	mechanics: Sebastian Blickle, electronics: Carsten Naber, software: Anton Schirg, organisation: Joachim Schoenmehl
Contact Email	mail@kamaro-engineering.de

Weight (kg)	Approx. 40kg
Actuators/motors installed	Dunkermotoren BG75x25CI 220W BLDC as main drive, 2x Dunkermotoren BG42x15 as steering servos
Turning radius (cm)	50
Battery voltage	6 LiPo cells at 3.6V = 22.2V
Battery duration (mins)	approx. 90
(W x L x H) (cm)	50 x 100 x 65
Sensors installed	LIDAR SICK LMS100, LIDAR SICK TIM551, 2x absolute encoder Pepperl&Fuchs CSS36M, 2x stereo camera Intel Realsense R200, fisheye camera, IMU BOSCH BNO055

Robot software description
The robot software is implemented on top of the ROS Software Stack. This means that the software is separated in so called nodes which solve small parts of the overall problem. There is a crawl row node keeping the robot in the middle between two rows of corn, a turn node which manages the switching between the rows and a localisation node providing the first two nodes with an accurate localisation to assist them. The localisation as well as a detection node using opencv empower the robot to detect golf bals. A state machine orchestrates the nodes to achieve the correct interplay for the given tasks.
Robot hardware description
Mechanical: In order to full fill the requirements of a robot driving in a field the drive chain was designed as a 4-Wheeldrive with a single, central electric motor which can provide a torque up to 9Nm per wheel. The power transmission flows on two self designed differentials in the front and in the back of the robot. Each axle mounting has its own suspension ensuring a smooth ride in rough terrain. Front and back axis can be steered independently therefore also diagonal movements are possible. **Electrical:** The central computing unit is a desktop computer mainboard in the mini-itx format with a Intel i5 quadcore processor and 8 Gb of ram. Low level hardware interaction and motor control via CAN-Bus is done on a ARM Cortex M4 Board. The microcontroller communicates with the computer via USB.
Task strategy description
We will use the LIDAR installed at the front to navigate between the rows of corn plants. For the weeding task we use a camera for detecting the golf balls and a 3D-printed construction for spraying. As our freestyle task we are planning to analyse the condition of the plants. We will use the stereo cameras to create a point cloud to measure their height or at least to differentiate the hights of different plants. Further we want to try to determine the health situation of the plants.

Carbonite

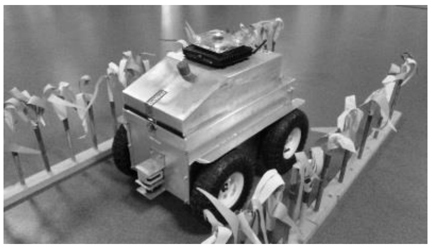


Name of Institution	Schülerforschungszentrum Südwürttemberg
Department	Standort Überlingen
Country	Germany
City	Überlingen
Webpage	sfz-bw.de
Team members	Jacob Schupp, Marcus Bolter, Christoffer Raun, Samuel Layer-Reiss, David Lippner (C), Timo Schönegg, Lukas Locher
Instructors	Lukas Locher
Contact Email	locher.l@t-online.de

Weight (kg)	16kg
Actuators/motors installed	4 Dynamixel MX 64 AR Servos (Steering). Robitronic Platinium 1:8 Brushless 10,5 T. Two servos and pumps for weeding unit
Turning radius (cm)	Not tested, theoretical 40cm
Battery voltage	16.8 V (4S Lipo)
Battery duration (mins)	Load dependent, minimal 240 min
(W x L x H) (cm)	45 x 78 x 30
Sensors installed	Pixhawk flight Controller (we only use gyroskope). Two Sick TIM- Laserscanner. Jaigo 5000 Cam

Robot software description
We use the robot operation system ROS (indigo) on an Intel NUC with Ubuntu 14.04 LTS operating system. For image processing we use openCV-library with ROS and Common-Vision-Blocks (CVB) from Stemmer-Imaging for the configuration of our camera. For parameter-adjusting of our ROS-Nodes we developed some GUIs based on QML.
Robot hardware description
We developped three PCBs. One for the power supply of all components. The other PCBs are based on stm32f0 microcontrollers, one for our weeding unit, the other is for speed-control, feeding the drive controller Toro TS 150 from sky RC with a PWM-Signal for desired speed. Sick-Laserscanners and NUC are connected over ethernet with the GIGE-Ports of our WLAN-Repeater. The Jaigo 5000 Cam is a USB3-Cam.
Task strategy description
We have two different approaches for navigating through a simple maize field. One approach is camera based - We use our camera to detect the maize rows and calculate the steering angle. The second approach uses two sick tim 2d laser scanners to detect the plants and to calculate a route to navigate through the rows. For task 2 we use our navigation tools from task 1 and our odometry tools to navigate between the rows. The weed and obstacle detection in task 3 is based on our camera as well. With the help of our obtained odometry values we create an opencv image showing the position of the detected weed in the field.

Dora the Explorer



Name of Institution	Fontys University of Applied Sciences
Department	Hightech agro minor
Country	Netherlands
City	Venlo
Webpage	https://fontys.nl/
Team members	Arjan, Luke, Jelle Stappers (Cap)
Instructors	Frank, Andy
Contact Email	j.stappers@student.fontys.nl

Weight (kg)	43
Actuators/motors installed	4
Turning radius (cm)	0 cm (skidsteer)
Battery voltage	24V
Battery duration (mins)	240
(W x L x H) (cm)	47 x 67 x 54
Sensors installed	4 encoders, 1 compass, motor drivers with temperature sensors, sick laserscan tim551.

Robot software description
The software of the robot is made in Labview.
Robot hardware description
the robot has 3 levels inside, the bottom level is where the motors are located, and being geared down, here are also the motordrivers installed to keep all Electromagnetic interference in the bottom level we grounded the chassis and split the robot up in levels. in the second level are located the batteries and Battery management systems (BMS). every motor runs on its own battery and BMS, to prevent voltage drop for the electronics or the pc. in the 3th level are the electronics for keeping track on encoders, power supply for the pc, laserscan and other sensors located. on top of the robot sits an i7 IPC2 to control the robot with labview, we have access to it over teamviewer.
Task strategy description
We plan to do Task 1 and task 2

FLORIBOT 2017

participants: Prof. Torsten Heverhagen, Benedict Bauer, Michael Gysin, Christian Scheuermann, Till Oetschger, Bernd Hückmann, Sergej Rommel

the developers: Manuel Zimmermann, Maximilian Veith, Maik Kheim, Dominic Bürk, Josua Epp, Dennis Hoch, and many more

Heilbronn University (HHN)

Introduction

The "Field Robot Event" project is becoming more and more popular at Heilbronn University every year. Especially since the victories at the 15th FRE in the year 2017, held at Harper Adams University, United Kindgom. Here the completely rebuilt FloriBot won the top position place in the task 1. Unfortunately for the second task it ran not quite as well. Here, the FloriBot reached the 5th place. Due to time-based design and focus, we had not finished the software for Task 3 and 4. Nevertheless, the FloriBot reached third place in the overall evaluation.

This elaboration provides a rough overview of how the FloriBot has emerged from many different studies and laboratory work.

Figure 4 Members at the FloriBot Team. From left to right: Michael Gysin, Sergej Rommel, Till Oeschger, President, Benedict Bauer, Bernd Hückmann, Christian Scheuermann

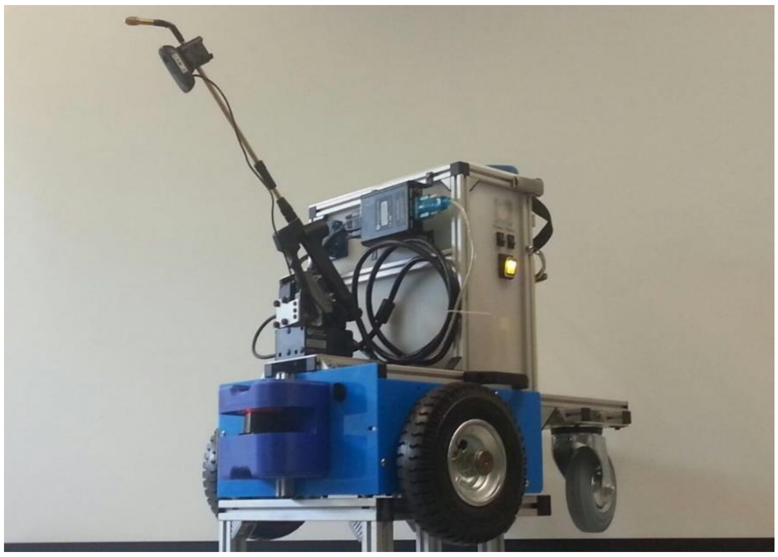


Figure 5 FloriBot 2017 in the new look

Overview

DIMENSIONS (W X L H) (cm): L 0.64m x W 0.50m x H 0.65m
WEIGHT (kg): 20 kg
ACTUATORS / MOTORS INSTALLED: 2 electric motors
SENSORS INSTALLED: Laser sensor all task, Cam for task 3,4
TURNING RADIUS (cm): on the spot
BATTERY VOLTAGE: 24 Volt
BATTERY DURATION (mins): unknown. Maybe 1h in use

Hardware

The FloriBot has been rebuilt in recent months. Within the scope of many studies, an almost complete new assembly of the robot took place. The most important aspect was the use and the optic of the robot. In addition, special attention was paid to lightweight construction. The new robot was built with light aluminum profiles.

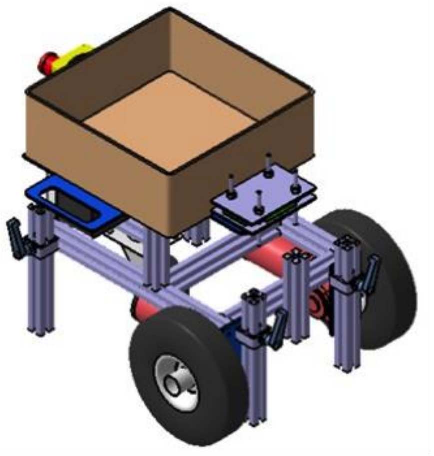


Figure 6 old FloriBot

It has a differential drive with two electric motors on the front axe. The motors are from ebm-papst and are BCI geared motors that are very powerful and silent. At the rear there are two additional swivel wheels installed. This allows FloriBot to turn on the spot. Both axes are rigid axis. So they are not feathered. The balance point is very deep since the motors are heavy. An up-and-down station was built for the robot as it has a very comfortable opening mechanism for maintenance.

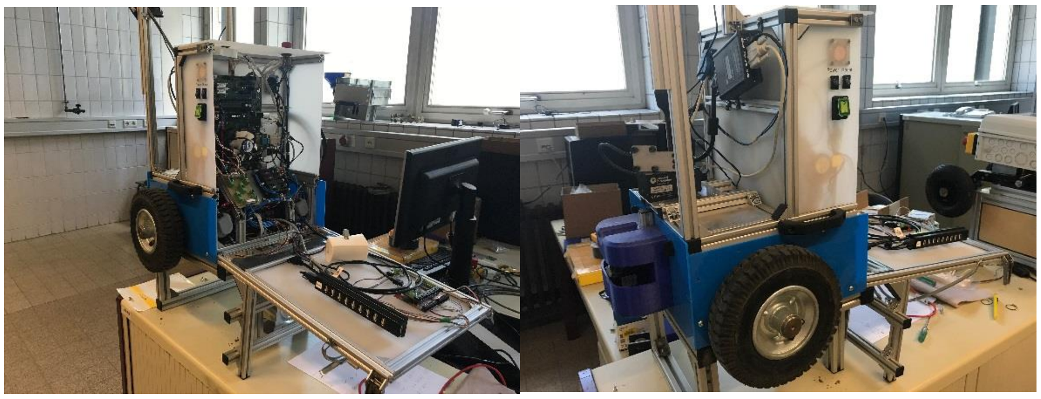


Figure 7 FloriBot open at the up-and-down station

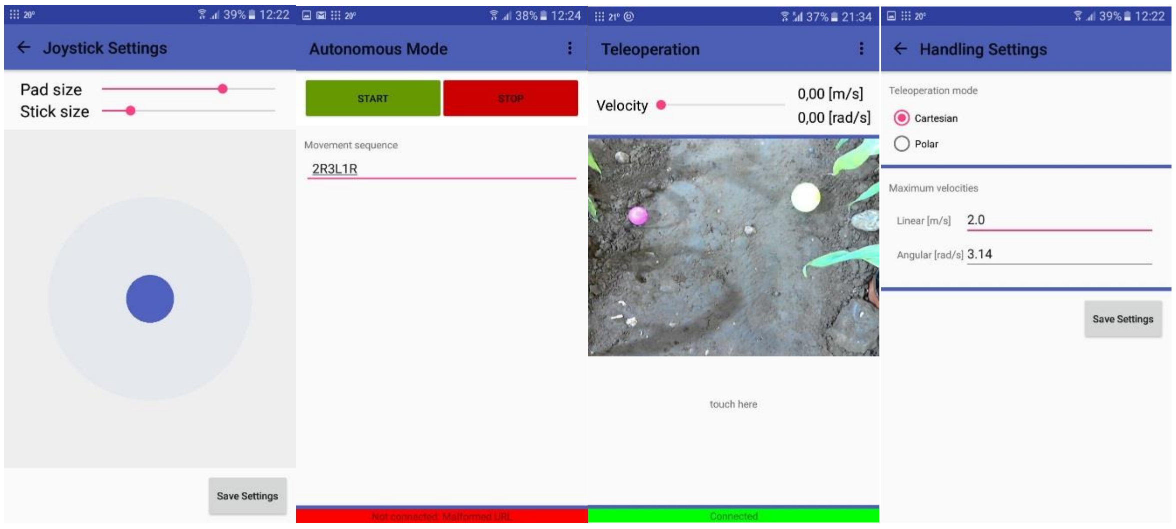
The FloriBot got a new very powerful motherboard. Otherwise it has the usual normal components for mobile robots. The FloriBot has a 2D-laserscanner in the front and scans the environment which enables the main orientation. For the front laser scanner we used Hokuyo. For Task 3 and 4 we used 1 Webcam to detect the balls.

Software

It was programmed with MATLAB / Simulink in combination with ROS. The programmed files are compiled since the FloriBot only works with C-code. The software architecture includes all important parts from kinematics to self-localization as well as an algorithm for traversing in the field.

Mobile App

The communication between FloriBot and human takes place through a mobile app and wifi. The mobile app was developed by master student Christian Scheuermann within the framework of his masters project. The app works on the Android operating system.



Movement sequence
2R3L1R

Not connected: Malformed URL

0,00 [m/s]
0,00 [rad/s]

Connected

Teleoperation mode
[✓ Cartesian]
Polar

Maximum velocities
Linear [m/s] 2.0
Angular [rad/s] 3.14

Figure 8 Different view from the App

Navigation

The same model is used for all tasks in the maze field. They only differ in the logic algorithm. The basic principle is based on the potential field method. The maze plants have a repulsive effect on the robot. A central scan determines the speed of the robot. In Task 1, we specify that a right curve should always be made after a left curve. In Task 2, we enter the pattern code via the mobile app. Task 3 and 4 are not finished yet.

Simulation

We have never tested the FloriBot on a real maze field at home. Under Windows operating system we used the simulation program V-REP and on Ubuntu Stage.

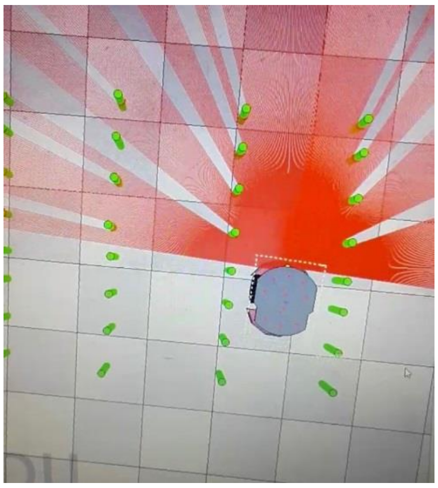


Figure 9 Simulation view in V-REP

The experience has shown us that when the simulated robot under Stage would behave well, then we asumed that it would do so in reality, too and wise versa. When he did well in V-REP he also did well in reality. But when he did bad in V-Rep he still did well in reality.

Suggestions for next year

- Make more tests at home in real cornfields.
- Rear axles
- New water tank with electric water pump for spraying task
- Airless tires from the 3D printer (Non-pneumatic Tires) (planning in progress)
- Better prepared for Task 3 and Task 4
    - Scan several rows at once
    - Swiveling syringe
- Use gyroscope sensor
- Install a LED lighting to see if it is autonomous or manual mode

To the Harper Adams University

The whole team FloriBot thanks from heart for the great event. It was very well organized from start to finish and extremely well looked after. Very pleasant was also that it took place in a hall.

Hefty

(Team Hohenheim)

David Reiser[1], Grischka Ulrich[1], Robert Hübner[1], Daniel Riehle[1], Hans W. Griepentrog[1]

[1]University of Hohenheim, Institute of Agricultural Engineering, Garbenstr. 9, D-70599, Stuttgart, Germany, dreiser@uni-hohenheim.de

Introduction

Robots could be the big invention of this century and allow machines to be more flexible and more agile to create and help humans in daily live. Field robots, helping producing the food of tomorrow could be a new part in the agricultural revolution taking part in the last two hundred years (Johnson, 1997). The Field Robot Event 2017 is once more a new opportunity to try to check out the abilities of robots, technique and programming abilities of student teams all over Europe. Moving in agriculture from big machines to small, autonomous machines can have a key role in solving actual challenges in crop production like soil compaction, machine transport, security or human labor costs. However, uncertainty in environment still makes it challenging to detect the values in noisy sensor data, like it is typical in agricultural sites (Hiremath et al., 2014).

Beside of just navigation, there is always a need for robotic research to show the use and ability for future applications. For this year, the authors developed a complete new robot system named “Hefty”, including mechanics and software. The whole system was programmed using ROS (Robot Operating System) (Quigley et al., 2009).

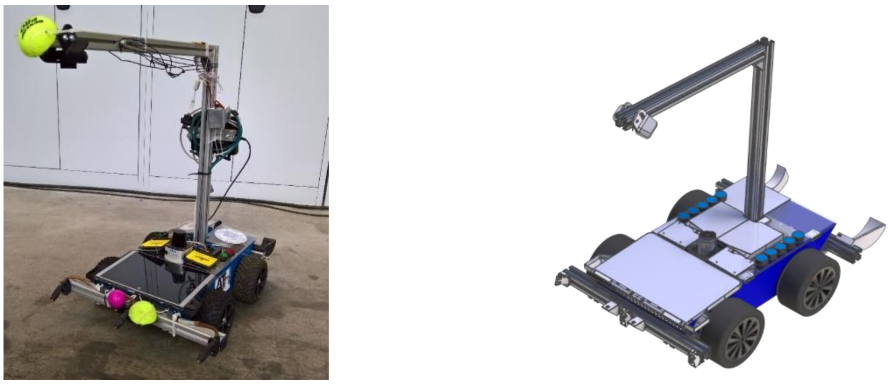


Figure 1: Robot Hefty: left in real, right as CAD visualization

Mechanics and Power

As basic vehicle, a small 4-wheel autonomous robot with differential steering was used (see Figure 1). The size of the basic robot platform is 500 x 400 x 150 mm. The weight of the robot is around 15 kg and it is equipped with four motors with a total power of 240 W. The used motors use a metal gear system of 1:50 what results in a speed of 200 rpm by 1.17 Nm per motor. The motors were directly connected to the wheels of the robot (see Figure 2). The motors at each side were connected with a belt system to ensure, that both motors at one side can pull with full force at one wheel. This ensured that at least two motors push whole robot forward. Maximum driving speed with the attached wheel diameter is 0.8 m/s and the maximum static motor torque of 4.68 Nm.

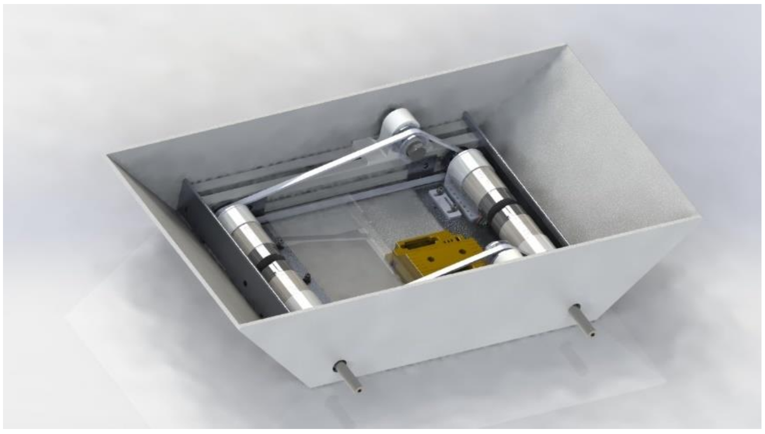


Figure 2: Mechanics of the motor belt system used in the robot Hefty.

For cooling the whole robot system, two computer fans were attached to the top, giving the option to work also under harsh conditions like in direct sunlight and hot weather conditions. The air gets sucked through the robot, by using an aperture at the front of the robot. This helps circulating fresh air inside the chassis. To avoid dust to enter, an additional air filter was attached. For the spraying task in the field robot event, an implement was attached to the robot, including a water tank and three magnetic valves. The magnetic valves were implemented at the front of the robot. The water tank was placed outside at the back of the robot, to avoid water damages.

The general Energy supply was provided by 4 LiPo Batteries, each with 11.1 V and 1000 mA. To be able also to attach a different Battery if needed, the robot was designed to be able to perform even with a lead accumulator, placed at the middle of the robot.

Sensors

The robot was equipped with different kind of sensors. Each motor was connected to an encoder (Pololu, Las Vegas, USA) with a 64 tick's resolution per motor rotation what corresponds to 3200 counts per rotation of each wheel.

For estimating the initial state of the robot, an Inertial Measurement Unit (IMU) was fixed to the frame of the robot. The used IMU was a VN-100 (VectorNav, Dallas, USA).

For sensing the environment, one light detection and ranging (LIDAR) laser scanner was mounted at the middle of the robot at a height of 0.3 m above the ground level. It was so mounted that the complete sight of 270 degree of the laser is usable. The used sensor was the TiM 571 2D-LIDAR from Sick (SICK, Waldkirch, Germany). The LIDAR was attached to a variable mounting platform, able to tilt and change the height of the sensor (see Figure 3).

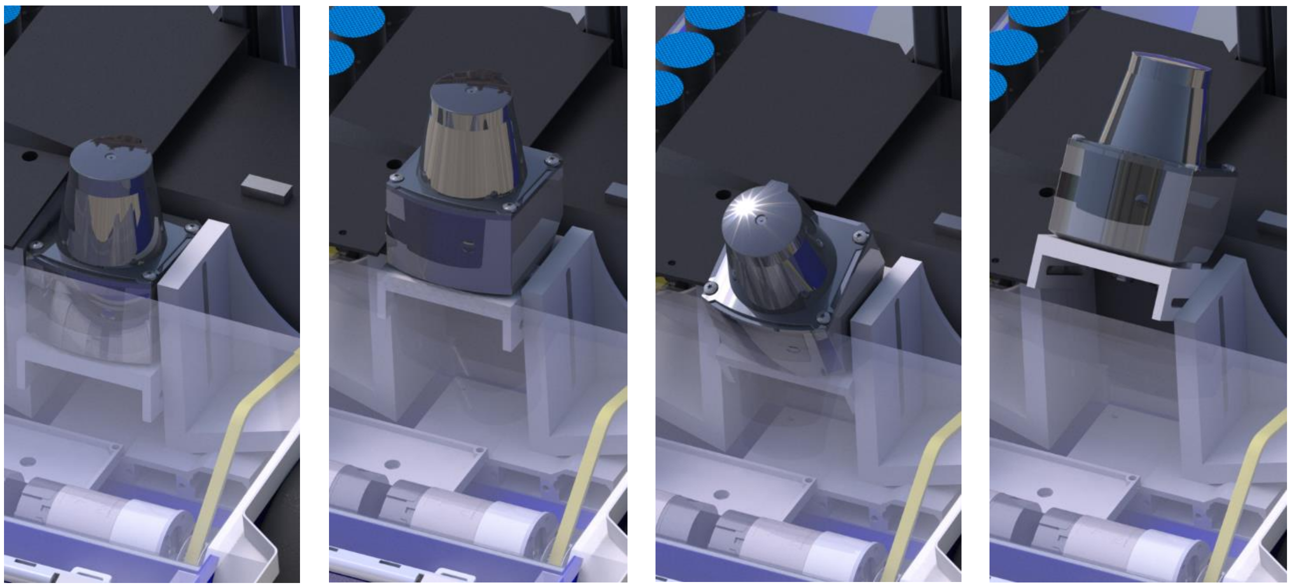


Figure 3: Attachment for the laser scanner in different positions

For detecting objects in front of the robot, an additional webcam was mounted at the robot, looking down at the ground. The used webcams was the Logitech C525 HD USB-Webcam (Logitech, Apples, Switzerland) The camera can perform with 8 MP but was used only with a resolution of 1080 x 720 pixels to speed up the image processing time. The position was defined that it was possible to detect the balls in front of the robot before they reached the spraying area of the attached nozzles.

Controller Architecture

Computer and Hardware

A tablet, connected to all necessary devices with a USB 3.0 Port, controlled the whole robot. The Tablet had an integrated battery with a runtime of 8 hours. It is using an Atom x5 1.44 GHz processor, 4 GB RAM and a 64 eMMC high speed storage. The touchscreen works with 1280x 800 pixels (10").

As motor controller the Roboteq SDC2130 (Roboteq, Scottsdale, USA) was used. This dual channel motor controller is able to power up to 30 V and 2x 20 A maximum, sufficient for the used motors. The motors at each side were connected to each other, so that they turned always in the same direction. In addition, the encoders were attached directly to the motor controller and the update signals were sent to the control tablet. For the remote control and direct user interaction a standard X-Box Joystick Logitech F710 (Logitech, Lausanne, Switzerland) was connected to the computer with a Bluetooth dongle. For activating the magnetic valves and the siren, four USB driven relays were used. The IMU used a RS232 protocol for communication, while the laser scanners worked with TCP/IP over Ethernet with an attached adapter.

For remote control a hotspot was integrated, enabling to control the computer tablet with SSH and remote desktop connection via WLAN.

Software and strategy

The tablet runs with Ubuntu Mate 16.04., including ROS as basic software structure for programming, visualization and user interface.

For competing at the field robot event, the most important task of the robot is the autonomous row following of the system. To set up this structure, a mode changer was implemented (Blackmore et al., 2002). This included one mode for navigation inside the row and one mode for headland turning. It was possible to overwrite the autonomous mode with a user joystick input and to separate the program code to different tasks. In general, three modes were used:

- User Mode
- Headland Turn
- In Row navigation

For each mode, the motor controller subscribed to a separate speed message, provided by the joystick, the laser scanner or the odometry with a point-to-point navigation. The ROS Middleware was used to set up the programs of the robot (Quigley et al., 2009). The whole programming was performed in using C/C++ and Python packages and nodes.

Task 1 & 2

For detection of the rows, the special object oriented structure of ROS was used to define a flexible and adjustable program, able to perform in different agricultural environments. So the start for the navigation algorithm were the raw data of the used LiDAR sensors. The aim was to define a goal point in the middle of the row, what was used to guide the robot for navigation. To reach this goal, several filter needed to be applied, to gain a position out of the data (see Figure. 4).

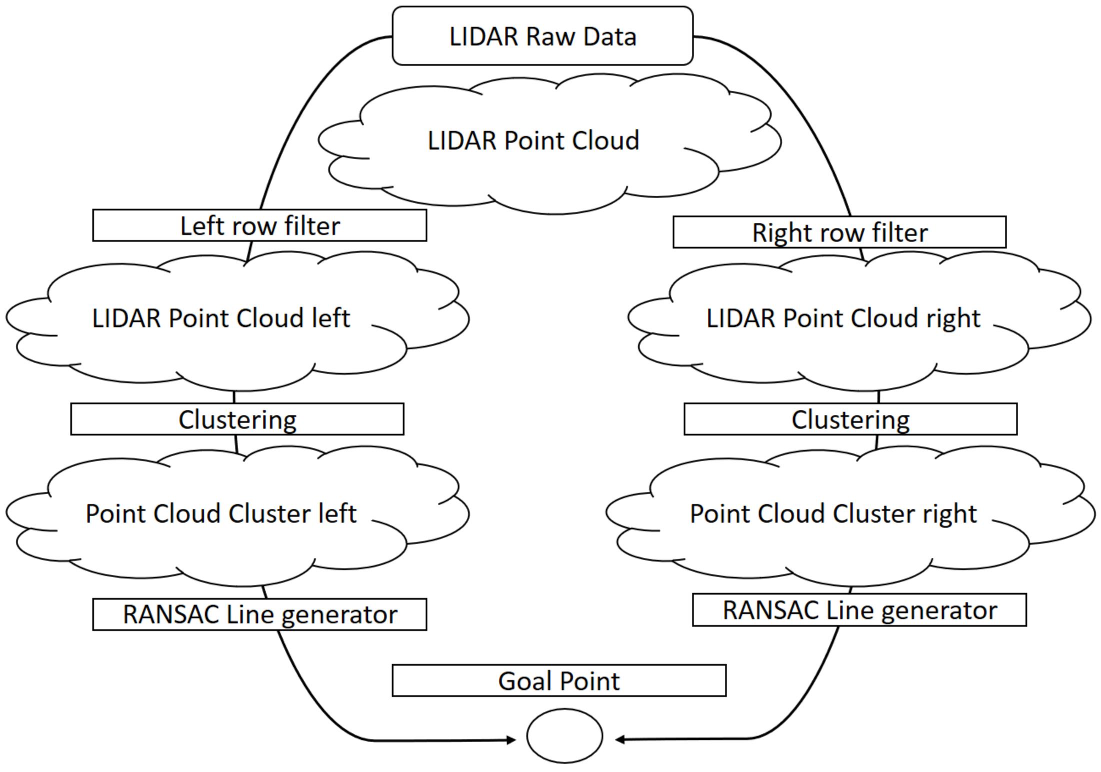


Figure 4: Description of the filtering algorithm used for the Field Robot Event 2017.

As the LIDAR Data was recieved in polar coordinates, they were first converted to a Cartesian koordinate system. To get rid of all unneccessary points, a box filter was applied, removing all points with a higher distance to the robot. So it could be assured, that just the two rows directly in front of the robot were detected with the sensor. Now the resulting points were separated in two different point clouds, one corresponding to the left row and one point cloud for the right row of the robot. Each of this point clouds were filtered, to remove noise out of the sensor data, bevore searching for the biggest cluster inside this selected area. The best fitting cluster was used to estimate the row on each side. Out of the two separated point clouds, the RANSAC row detection algorithm could be applied to each row separate. Out of the difference of both lines, a navigation point was estimated, taking the orientation and the position of the lines into account. Figure 5 is showing the point cloud of the front laser in combination with the base coordinate system of the robote (base_link). The next goal point (row_point) coordinate system is defined out of the two filtered point clouds and the resulting lines estimated by the RANSAC line detector.

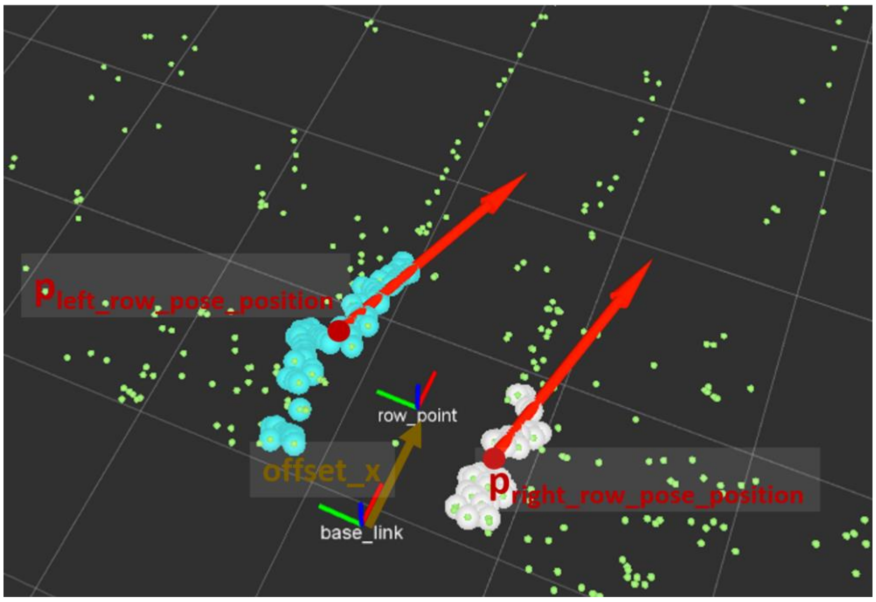


Figure 5: Description of the rows and the resulting row point. The base_link coordinate system corresponds to the robot center.

The change for the mode from row navigation to headland turning was performed when no points in a distance around 1.5 m ahead of the robot were detected.

The headland turning used goal points to estimate the next position for the robot and a point follower addressed the necessary speed signal. The turning was performed with three goal points. As soon as the last goal point was reached, the mode changed again to row navigation. For evaluation of the algorithm, the sensor outputs and filtered data could be shown in real-time in the RVIZ visualization tool (see Figure 5).

As soon as the headland was detected, odometry was restored and resolved by the use of wheel encoders and the IMU data to move to the next row. The headland turns were defined by a list of row points, what could be changed for every single turn. So the same program for task 1 and 2 could be used. The only changes were made in waypoint lists before the start of task 2.

1.1.1 Task 3 & Task 4

To perform this task, the navigation algorithm from task 2 was used, including a 180 degree turn in the row, when an obstacle was detected. The pink golf ball was detected and sprayed by a simultaneous running task.

The detection of the golf balls was performed by one webcam, placed at the front of the robot. It was arranged to detect the full row area in front of the robot with 75 cm width.

The camera used 3 areas to detect the ball and was giving out a topic when in one of the areas a ball was detected. This signal was directly forwarded to the magnetic valves, activating the spraying at each of the 3 spots. This helped, that the spraying system did not need any information about speed to perform, beside a vehicle guiding it through the crop rows.

The image analysis was performed, using OpenCV. The pink golf ball was filtered with a min and max value of Hue (H), Saturation (S) and Value (V) of the color, after converting the image to the HSV-color frame (see Figure 6, left). Out of the result, a binary image was generated (see Figure 6, right).

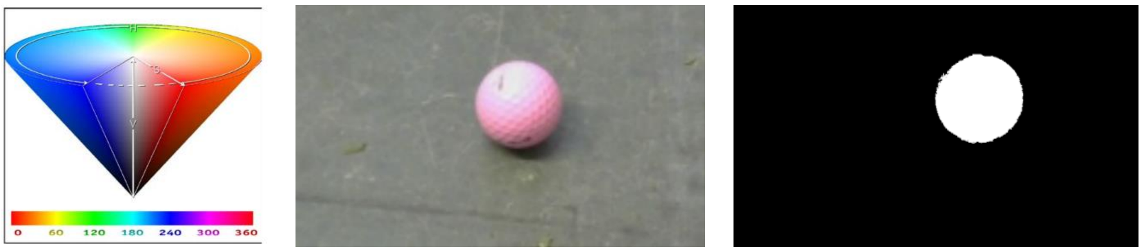


Figure 6: HSV Color space left, RGB image middle, binary image of a filtered image right.

Besides just using a fixed color space, we used a reference color, of a fixed spot at the robot to guarantee that lightning conditions will not affect the ball detection even under changing conditions while runtime. For that, a detection zone was defined in the image and an analysis area, where the color limits for the ball were automatically solved (see Figure. 7).

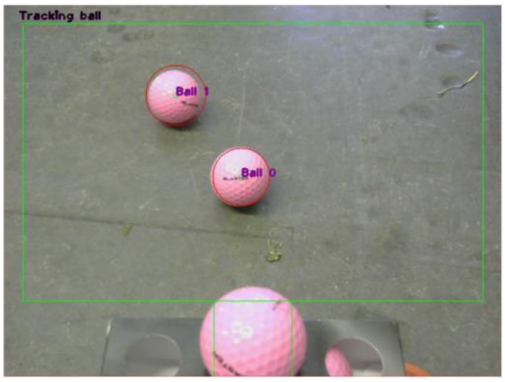


Figure 7: Area of the balls with automatically color limit definition.

The whole algorithm for processing the binary image is shown in the following flowchart of Figure 8. First, the raw image was converted from RGB to HSV color space. Out of the reference ball (see Figure 7) the range for the HSV color min and max was defined, what was used to filter out a binary image out of the color image. For filtering there was a blur filter applied, helping to get rid of small pixels in the field. Afterwards the Hough circle algorithm was applied to the binary image, resulting in a number of possible circles detected in the field. They were filtered for size, to check out if the result are realistic or if the detected ball was just noise. This helped to decrease the number of false positives. As soon as the ball could be confirmed, a Boolean message was send to the relays to activate the corresponding valve.

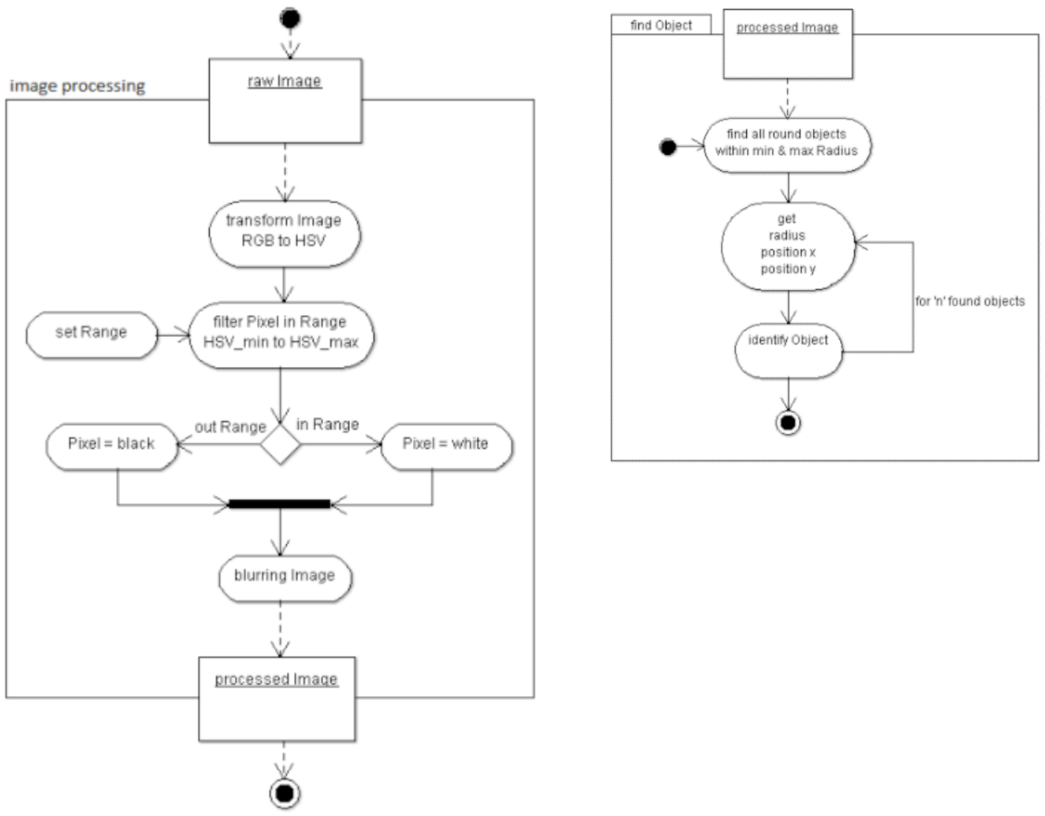


Fig. 8: Left side: Filtering of binary image; right side: the algorithm for identifying balls in the binary image.

The following Figure 9 is showing that light did not affected the outcome of the algorithm and the robustness of the used method for changing lightning conditions.

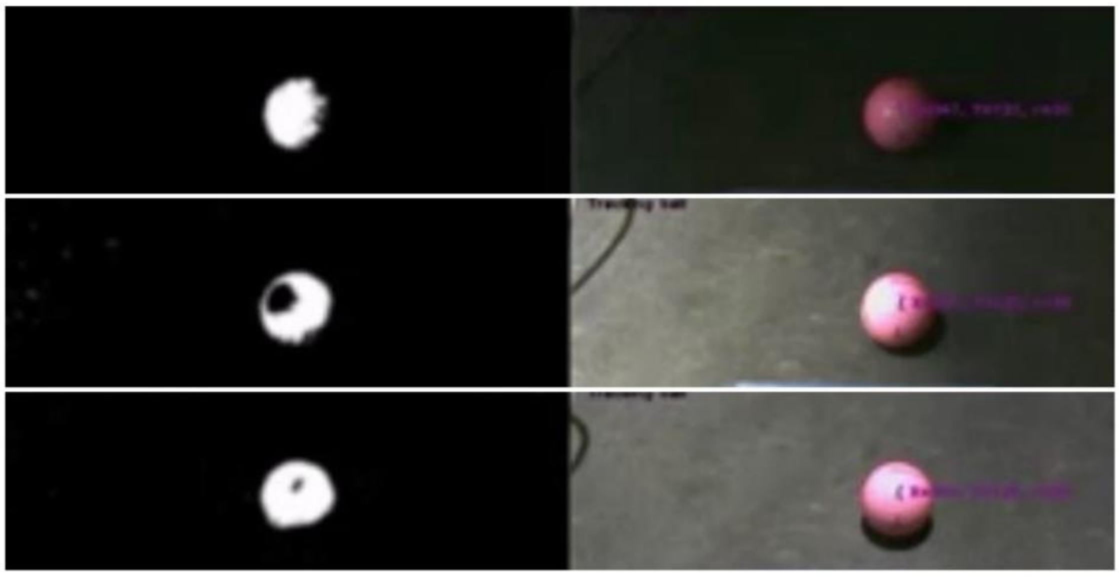


Figure. 9: Algorithm outcome under different light conditions with automatic adjustment.

Conclusion

This year the team Hohenheim competed with a new robot called Hefty and was able to perform with this robot the first 3 tasks even under the problematic conditions of the rough terrain of the test field. One big issue was the weakness of the motors compared to the weight of the robot. Possibly losing weight of the robot or the integration of different wheels could help. Also a more suffocated hotspot and user interface could help to perform better next year tasks. An earlier possible testing of the algorithms with the final vehicle could also help to perform better at the competition.

However, this year was a great success for our team, we looking forward to compete next year again and using the gained knowledge for better results.

Acknowledgements: We want to thank Claas Foundation, Sick, Mädler and the University of Hohenheim for sponsoring our team.

References

Blackmore, S., Fountas, S., Have, H., 2002. A proposed system architecture to enable behavioural control of an autonomous tractor., in: Zhang, Q. (Ed.), Automation Technology for Off-Road Equipment. 2950 Niles Road, St. Joseph, MI 49085-9659, USA, ASAE, pp. 13–23.

Hiremath, S.A., van der Heijden, G.W.A.M., van Evert, F.K., Stein, A., ter Braak, C.J.F., 2014. Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter. Comput. Electron. Agric. 100, 41–50. doi:10.1016/j.compag.2013.10.005

Johnson, D.G., 1997. Agriculture and the Wealth of Nations, in: Papers and Proceedings of the Hundred and Fourth Annal Meeting of the American Economic Association. pp. 1–12.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Mg, A., 2009. ROS: an open-source Robot Operating System, in: ICRA Workshop on Open Source Software. p. 5.

HELIOS
FREDT

Andrii Kostrytsia, Christian Schaub, Christopher Prange, Christopher Sontag, David, Bernzen, Enrico Schleef, Henrik Wulferding, Julius Steinmatz, Luke Schneider, Matthias, Kemmerling, Maximilian PleB, Mohammad Al Zoubi, Sven von Hoveling, Tom Schroder

1. General:

Helios has been developed by our team for ten years now and still has a very feasible design for mastering the challenges of the Field Robot Event. It stands out in robustness and usability, since it's a straightforward, simple construction which fulfills every major functionality needed.

2. Mechanics:

The durable Helios drivetrain realizes a permanent four-wheel-drive and consists of an electric motor (Dunkermotoren BGT5X25CI, 250W), a set of gears and axle differentials and four wheels with equal size. The gears and differentials are separated from the rough and dusty environment through aluminium housings and the damped axle construction can easily adapt itself to very uneven ground. Therefore the robot is ideal for the use under often unpredictable field conditions.

The vehicle uses an Ackermann steering concept at both axles which combines a small turning radius and smooth movement at the field headland with large vehicle stability when driving straight between the crop rows.

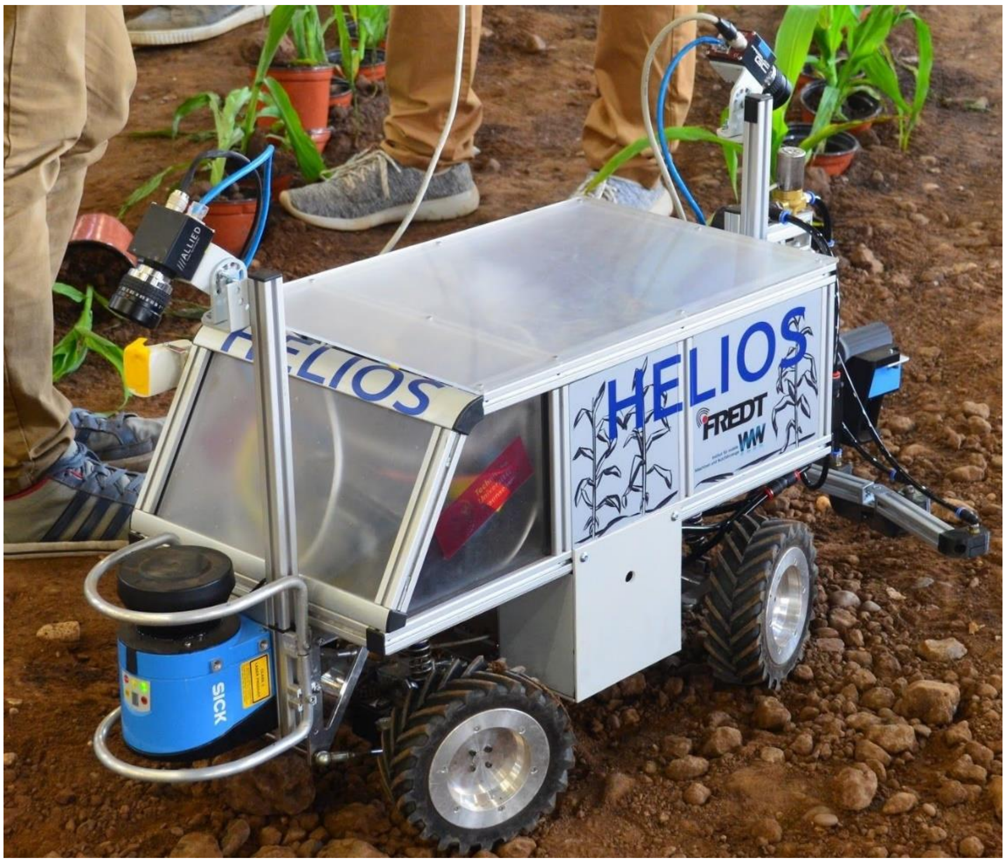
3. Electronics: Nickel-metal hydride battery with 24 volts, remote control, GIGABYTE Barebone

Sensors and Protocols: Laserscanner SICK, Router, Switch, CAN, Ethernet

Laserscanner:

The most important robot sensor is the Sick LMSlOO 2D laserscanner, with a maximum range of 20m and 270° observation angle with 0.5° angular resolution at 50 Hz scanning frequency. It is permanently installed at the front of the vehicle and provides a precise capture of the near and far environment.

Additionally, there is the much smaller Sick TiM31O laserscanner (4m maximum range, 270° angle, 1° angular resolution and 1Hz). It is attached to the rear of the vehicle both in horizontal position to support the data from the front laserscanner.



Helios inside field, showing laser scanner and camera.

4. Software:

We are using ROS, the robotic operating system, an open source environment for controlling robotic platforms. Its speciality is the heavy usage of the observer design pattern which, beside other techniques, decouples the need of thread and inter-process-communication management for the developer and leaves more room for the important parts.

<u>Approach task1</u>

Task 1, basic navigation, was aimed at an autonomic navigation of the robots through long rows of corn plants. At the end of each row the robots had to turn and return in the adjacent row. Therefore we implemented a simple controller. It evaluates the data from our front laserscanner which detects the distance between Helios and row and initiates the necessary steering angle to correct the driving direction. 80 the shorter the distance to the right the bigger is the calculated steering angle to the left and vice versa. To navigate inside the row we used a normal front wheel steering powered by servos and for the turn at the end Helios has a four-wheel control.

<u>Approach task2</u>

Task 2, the advanced navigation, resembles task 1. The difference is that the robots have to drive through the rows in a given order. We solved the navigation through one row similarly to task 1 with our simple controller. The navigation from the end of one row into the next given row was the difficult part. We solved this by writing a mathematical function that calculates an ideal path depending on the given order. Our robot uses this function and tries to follow the path as exact as possible using his odometry.

<u>Approach task 3</u>

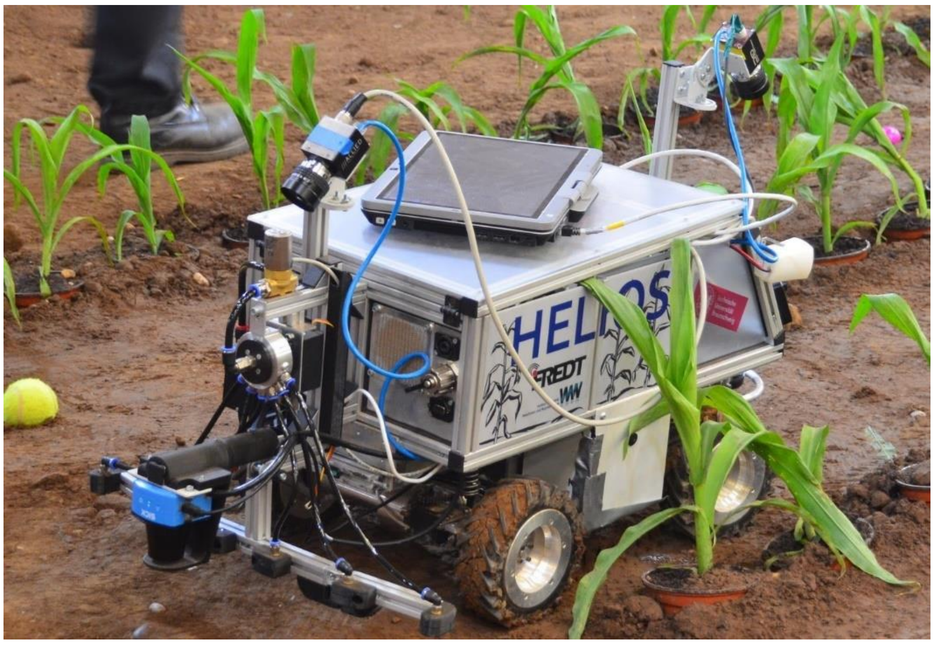
The idea of task 3, mapping, was to detect purple golf balls in a field and to log them in a map which will be used in task 4. Furthermore, yellow tennis balls were placed inside the field, representing obstacles which must not be passed. An audible signal was supposed to show that the robot detected a golf ball or a tennis ball.

Our robot used a front and a back facing camera in order to detect the balls. The detection was accomplished using OpenCV and by looking at the average colour values of detected balls. Taking the ratio between distinct RGB colour values, we were able to distinguish between the yellow and purple ball. A small siren produced a short signal for golf balls and a long signal for tennis balls. The position of the balls were logged into a file with their corresponding x and y values. On encountering a tennis ball, our robot was able to turn its direction and drive backwards out of the row, continuing the mapping while driving backwards until encountering another obstacle.

Approach task 4:

This task required the map which was generated in task 3. Using the map, the teams had to figure out an optimised path through the field to spray the weeds. An area of 25 cm diameter around the weeds had to be sprayed with water.

Our team built a module which allowed us to spray water out of one of several nozzles which were attached to a vertical profile. On this profile, the nozzles were spaced along the width of the row allowing us to spray the weed with the required precision. A servo controlled the flow of water, directing it to the nozzles. The position of the balls and the corresponding position was calculated with the help of the cameras.

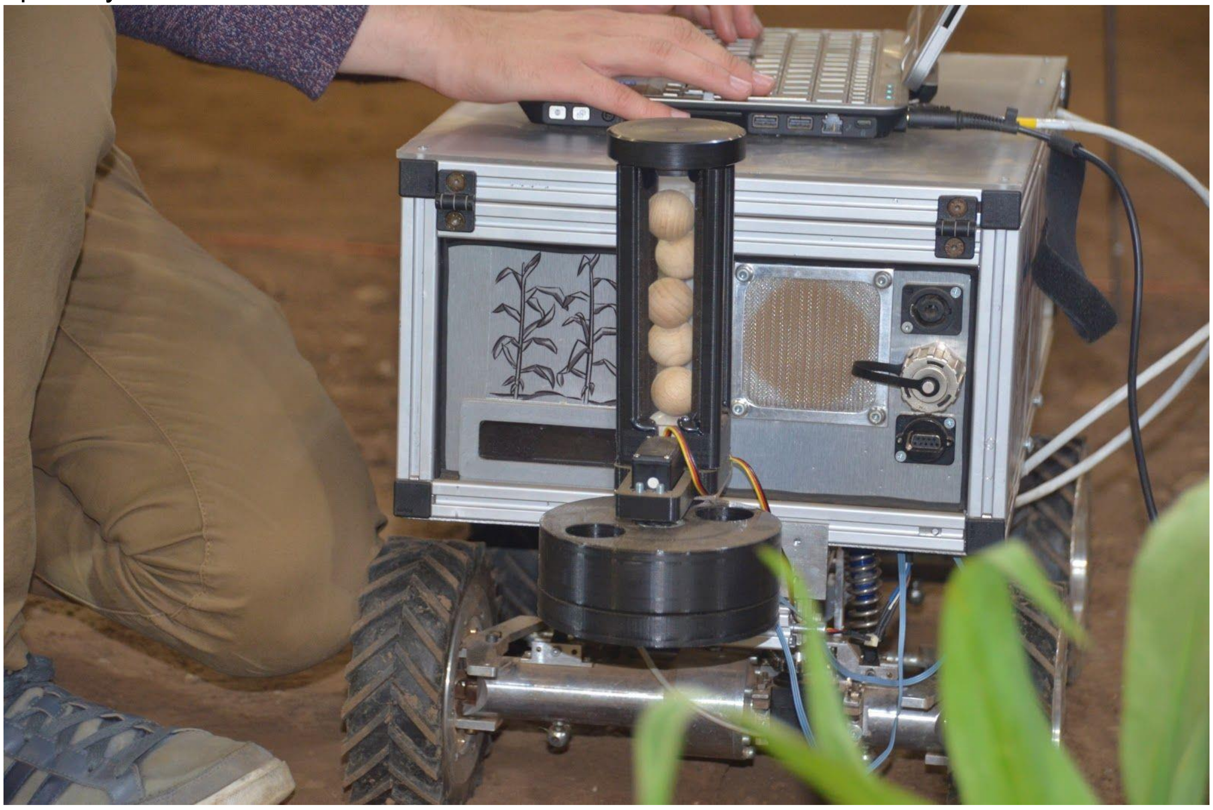


Helios inside the field with cameras, siren and spray module.

Approach task 5

For task 5 our team tried to find a method of deploying seed pods of scorpion wasps. These wasps are a common enemy of the european corn borer, a parasite that destroys maize plants, costing the german agriculture industry more than 11 million €. The seed pods contain a multitude of wasp eggs which hatch into larva, killing the corn borer.

The pods are currently either deployed manually or via drone and we designed a way to deploy the seeds using our field robot. The module uses a servo to drop individual pods after a set driven distance in order to optimally distribute them inside the field.



Module to deploy wooden balls which are representing seed pods.

6. References:

Our website:

http://wwwfredtdef

Freundes- und Forderkreis des Instituts ftir mobile Maschinen und Nutzfahrzeuge e.V

https:f/www.tu-braunschweid.de/imn/foerderverein

K9

Weight (kg)	50
Actuators/motors installed	8x whiper motors of opel corsa (Vauxhall Nova)
Turning radius (cm)	0
Battery voltage	12V
Battery duration (mins)	Unknown
(W x L x H) (cm)	55 x 60
Sensors installed	xv-11 (neato), Xbox kinect 360, pixycam, gy-80 (IMU), induction sensor, AS5048B

Robot software description
ROS for navigation , arduino for lowlevel control of the motors
Robot hardware description
1x salvaged i7 laptop for running ROS, 2x Arduino Mega's for motor control, 8x vnh2sp30 motor controllers for running the motors, 4x Free sample AS5048B magnetic angle sensor for steering positioning, 4x salvaged induction sensors as motor speed encoder, 1x GY-80 IMU as extra reference, 1x Pixycam, 1x salvaged Xbox kinect 360, 1x xv-11 Lidar (salvaged out of an Neato botvac vacuum cleaner), 4x salvaged bicycles for the steering mechanism and used for building the frame, 4x 28cm wagon wheels, 1x 12V 24Ah battery salvaged out of an UPS system
Task strategy description
Pre-task mapping with kinect sensor. Navigation with ROS via Full RGBD-SLAM with IMU and Laser range finder.

TAFR

Janez Cimerman(1), Iza Burnik(1), Žiga Brinšek(1), Gal Pavlin(1), Tim Kambič(1), Pavel Remic Weiss(1), Martin Debevc(1), Petra Bergar(2), Maša Florjančič(2), Matej Avšič(3), Rok Avšič(3)

(1) Faculty for Electrical Engineering, Ljubljana, Slovenia
(2) Faculty of Natural Sciences and Engineering, Ljubljana, Slovenia
(3) Faculty of Chemistry and Chemical Technology, Ljubljana, Slovenia
Ljubljana, Slovenia, 2017

Introduction

One of the disciplines that is on the verge of technological leap is agriculture. We can already see real development on technologies from smart surveillance with drones to small ground autonomous robots working in fleets. All these have recently been made possible due to rise of open-source technology and software, cheaper sensors and actuators and the need to automatizate work on farms.

Project TAFR was started to advance the knowledge and awareness of automation technologies in farming among students and general public. This robot was designed to tackle real world farming scenarios: driving in between rows of corn, check for diseases and spray them with pesticides (all autonomously). Although diseases were represented with clear markers (purple golf balls) and robot was only spraying water, the possibility of the automatization of work on fields was demonstrated with great success.

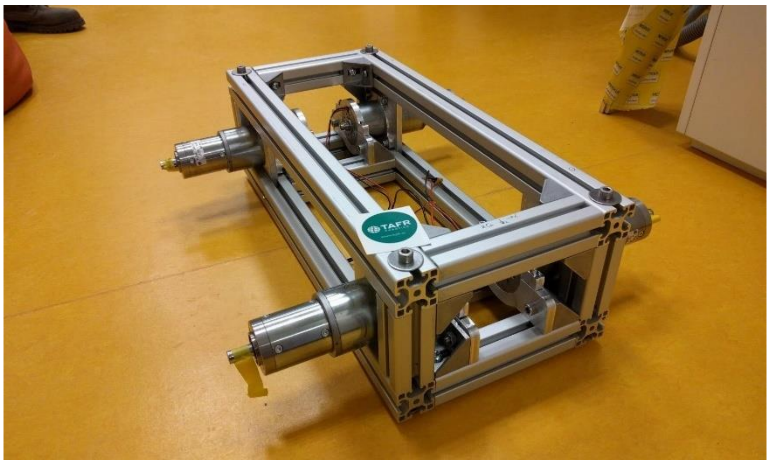
This report briefly describes the technical aspects of the project, as well as problems and solutions faced when designing the robot for farming.

Mechanics and Power

The main guideline for building the whole robot was simplicity. The robot was built with minimal, simple parts, which are easy to replace and/or modify. This gave us a real advantage when testing and correcting all components on the go.

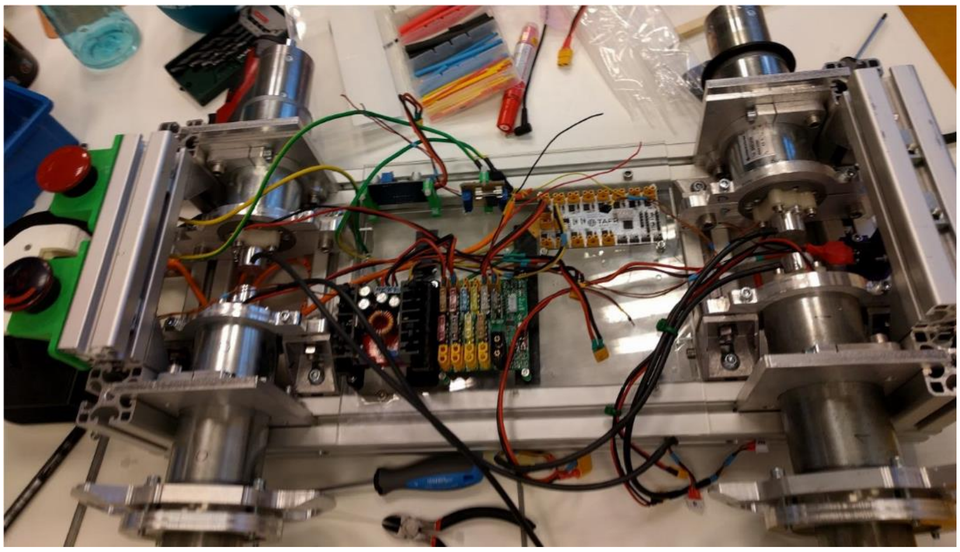
Robot’s Mechanics

Robot’s chassis is constructed from 40mm extruded aluminium profiles, because they are the easiest and the cheapest to construct and later modify. The mounts for motors and mounts for tires were all custom milled for this robot as were the Plexi plates on the side of the robot for dust protection. The tires are mounted directly on the motors and connected directly to the motor’s axis to achieve extreme simplicity and rigidity of the design.



Robot’s Electronics

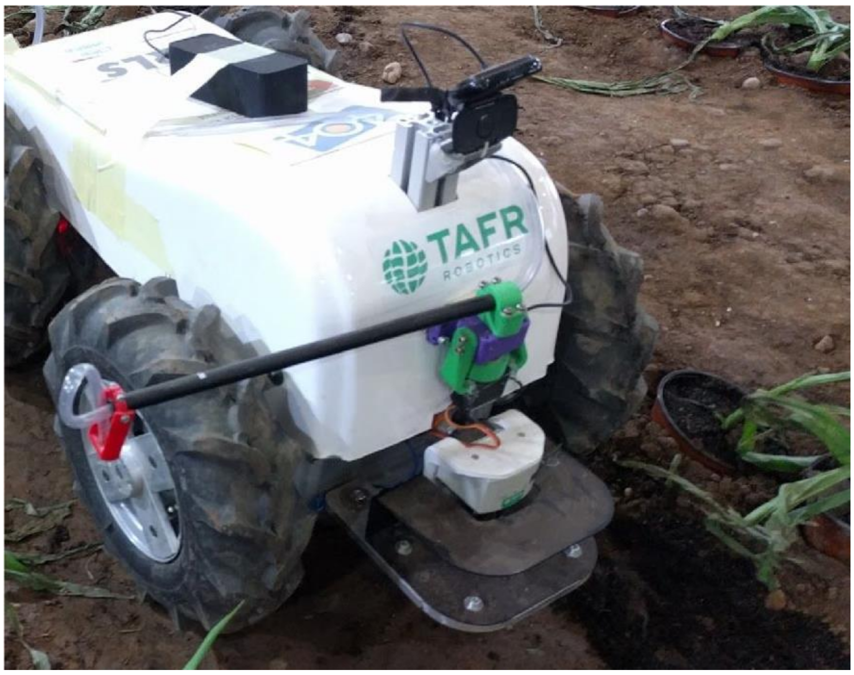
The whole robot gets power from 24V 10Ah LiPo batteries stored at the bottom of the robot. The main power consumption happens in motors, which are controlled by RoboClaw motor driver from Pololu. We also installed fuse board (made ourselves) for the protection from high currents and peripheral board (also made ourselves) with which we control all the additional electronics (for example: sprayer). The central computer is Intel NUC which has i5 processor and 8GB of RAM. Other parts of electronics also include various voltage step-downs, rotary encoders on each motor, SICK Tim LIDAR and it’s control board, IMU,... We have two safety switches at the back of the robot. One cuts power only to motor drivers and the other cuts power to the whole robot and is used regularly also as ON/OFF button of the robot.



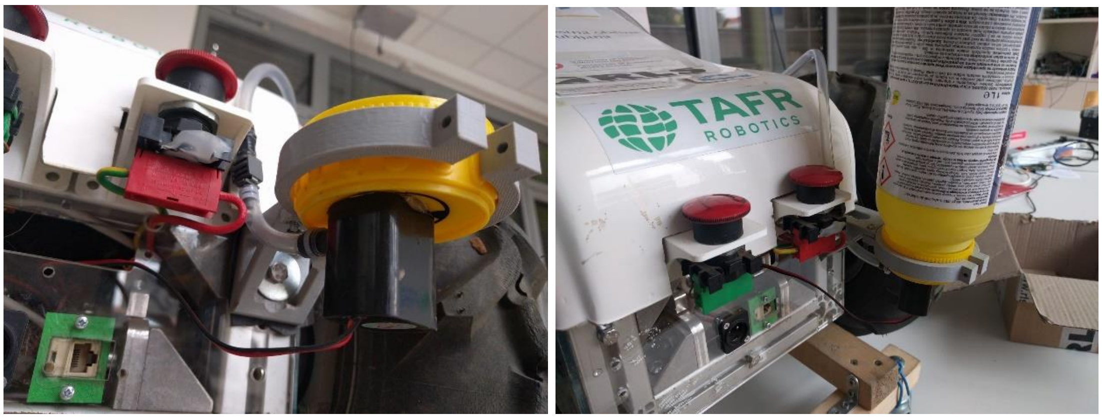
The robot has two electricity power inputs. The first one is from LIPO batteries and it is used as main power source for robot while moving. The second one is on 12V connector at the back of the robot and is used only to power computational electronics (NUC and all sensors). We use the second one during stationary tests (calibrations, algorithms development) so we don’t have to change the battery all the time. We also use it for changing the main battery, so all our systems stay turned on and we don’t have to restart them.

Sprayer

The sprayer we used consists of two main parts: the front and the back. The front is a simple mechanism with servo and carbon-fiber tube, used for aiming the water jet at the target (disease in the row of corn represented by purple golf ball).



The back of the sprayer module is comprised of bootle holder, the bottle and a 12 volt pump. The bottle is turned upside down with the pump on the bottom to keep it constantly flooded (otherwise it does not pump out water). We also designed a simple one way stream tube, to keep the water from flowing backwards to the bottle.



Controller Architecture

Computing Hardware and Sensors

All of the computing is done on a single Intel NUC. The main sensor we used for navigation is lidar from Sick (10 meter range, 0.33° resolution). For turning on the spot we used IMU from Bosch (BNO055) and for all the computer vision the Logitech’s C525 web camera was used.

Software strategy

3.2.1. Task 1

For following the rows of corn we used the following algorithm: first we performed clustering on lidar scan data, which returned us approximately 1 cluster per plant. Than the closest cluster to the robot was found and the distance to that cluster was used as reference for PID controller used to calculate angular velocity of the robot. For linear velocity the same distance was used but this time with only proportional controller.

When the end of the row was detected the robot drove straight for 0.7 meters based on odometry from encoders positioned on motors. At that point the robot turned for 90 degrees in the desired direction. The turn was controlled with data from onboard IMU sensor (gyroscope+magnetometer).

After the turn the robot used the same clustering as in rows to follow the end of the rows. For counting the passed rows the difference between two consecutive distances to closest cluster was calculated and when it surpassed the specific threshold the row count increased. When enough rows were counted (in this task just one) the robot again turned 90 degrees to face the plants. To drive in the row the same algorithm as for row following was used, but with more aggressive PID values.

3.2.2. Task 2

Exactly the same algorithm was used as in first task. Only the numbers for counting rows were changed.

3.2.3. Task 3

In this task we used the same algorithm as in first task to drive the robot in every row. For detecting colored balls representing diseases and obstacles we used USB web camera. Firstly we used color segmentation to get the position of the ball. The position (in pixels, relative to camera) was than transformed to reference coordinate frame of the robot using homography. Using that information and information from odometry the global coordinates of the ball was saved to separate file. At the end of the task we used separate python script (and OpenCV) to visualize saved data on a map.

3.2.4. Task 4

For spraying task the same base code was used as in second task which enabled us to load an optimized path for the robot to follow. When the robot was close enough to the disease to be able to spray it, the robot stopped. From visual data the correct angle for the sprayer was calculated and the disease was sprayed. We also implemented simple software to track the balls so we didn’t spray the same ball twice.

3.2.5. Freestyle

Farmers tend to use pesticides more carelessly than is considered healthy. Due to lack of protection when in contact with them farmers often get exposed in ways from direct spray, drift or contact with pesticide residues on the crop or soil. The use of robot to limit the exposure is a great leap towards reducing exposure, but we wanted to go one step further.

We developed a smart system for robots, that makes it possible for farmers never even to have a possibility to be exposed to harmful pesticides.

The system recognises which product you have by scanning it with the front camera and searches its database to find all data about it. All the important data, like suggested mixtures, safety hazards, and instruction of use, are shown to the user through a web interface or embedded screen on the robot. Through this interface user can adjust the parameters according their own preferences.The user then puts the bottle with pesticides on the robot and presses apply. The robot mixes the pesticides with water and applies the mixture on the field all autonomously.

This way we not only prevent farmers to being exposed, we also ease the use of pesticides by giving the user the right information at the right time.

Conclusion

After one year of developing autonomous farming technologies we can conclude that the problem isn’t as trivial as it seems. All of the unattended problems in hardware affects the development in software and vice versa. The problem also resides in being in uncontrolled outdoor environment. All in all with superb technology all that problems can be overcome and solutions for autonomous farming made available to the public.

Partners

We could not build this robot without the help of our partners:

General partner: Zavod 404

Research partner: Mattro mobility revolutions

Golden partners:	Laboratory of modeling, simulation and control
	SICK
	Epilog
Silver partner:	Laboratory for telecommunications, FE UL
Bronze partners:	RLS
	Agromehanika
Project supporters:	Mestna občina Ljubljana
	MakerLab Ljubljana
	RRA LUR
	MiranKambicPhotography

Terra

Zhuo Cheng, Laiquan Luo, Shenghui Yang, Yanqiu Yang (Female)*

*Team Leader & Paper Author

Harper Adams University, Newport, Shropshire, England

The International Field Robot Event (FRE) 2017 was held at Harper Adams University – 14 teams from countries including the UK, Netherlands, Mexico, Slovenia and Finland competed in the university's Soil Hall, an indoor field.

The FRE has been held annually since 2003, which provides new visions to visitors including farmers and engineers. We are a group of Chinese students who are currently studying Applied Mechatronic Engineering at Harper Adams University. With a great passion to learn new technology, we participated in this event, which indeed opens a window for us. The team would like to thank the Douglas Bomford Trust for their financial support for entering the event and the university staff for their technical support.

Figure 10 debugging the robot

Robot Platform

This robot is an "off the shelf" radio controlled (RC) vehicle, which provides a reliable, powerful and functional platform for the mounting of mechatronic parts. Figure 2 shows an exploded view of the sensing components on the vehicle, from which the LED scanners (1) can be seen. Three of these were used, one in front to guide down rows and two on the side to detect row ends and navigate in the headland. Two digital imaging cameras were mounted to the front (2) which both detect weeds in task 3. These were mounted on laser cut steel (1mm) and had polystyrene covers to prevent water ingress. A compass was mounted on a mast towards the rear of the vehicle (3) to indicate vehicle direction.

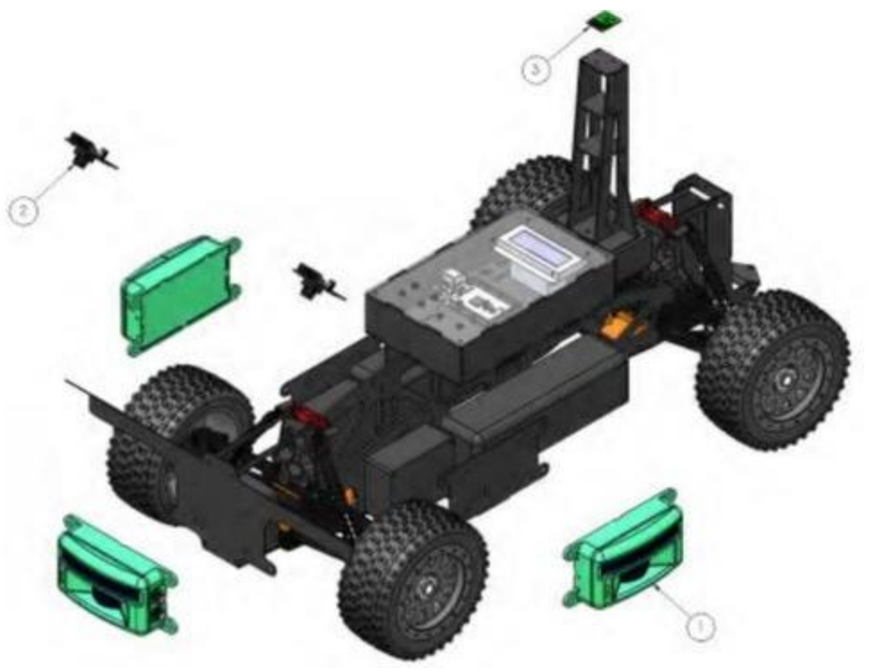


Figure 11 exploded view of the vehicle

Two 11.1V 3S 3500mAh batteries provided all of the power required: one supplied the high current components (drive motors, sprayer pumps etc.) whilst the other supplied the control and sensing components.

Control System

A system of distributed control was used, which allows each controller to take care of time critical sensing (such as reading of the encoder) and then pass data on request to the system controller. This was implemented so that a lower level of controller could be used – microcontrollers in place of an embedded computer. For this application, benefits such as a lower cost; lower software complexity and easier hardware interfacing could be achieved. Controller communication occurred over an i^{2}c bus with the system controller as the master. Figure 3 details the controllers used on the robot and their connections.

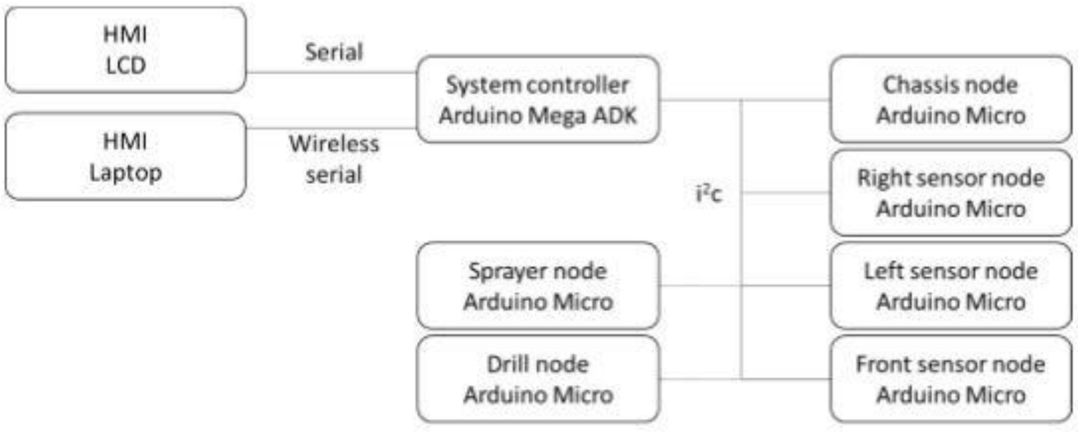


Figure 12 complete controller system

Power, at a level of 11.1V, was provided to all on-board controllers (omitted from figure 21 for clarity). Figure 4 shows the Arduino Mega ADK used as the system controller and the Arduino Micro used for each of the nodes.

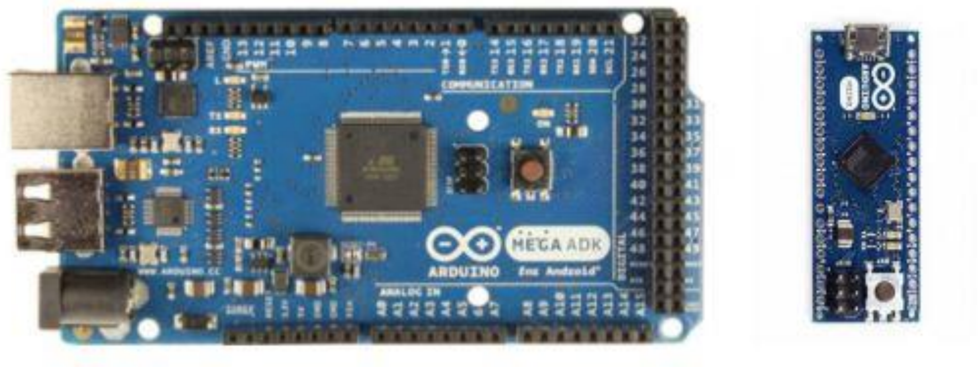


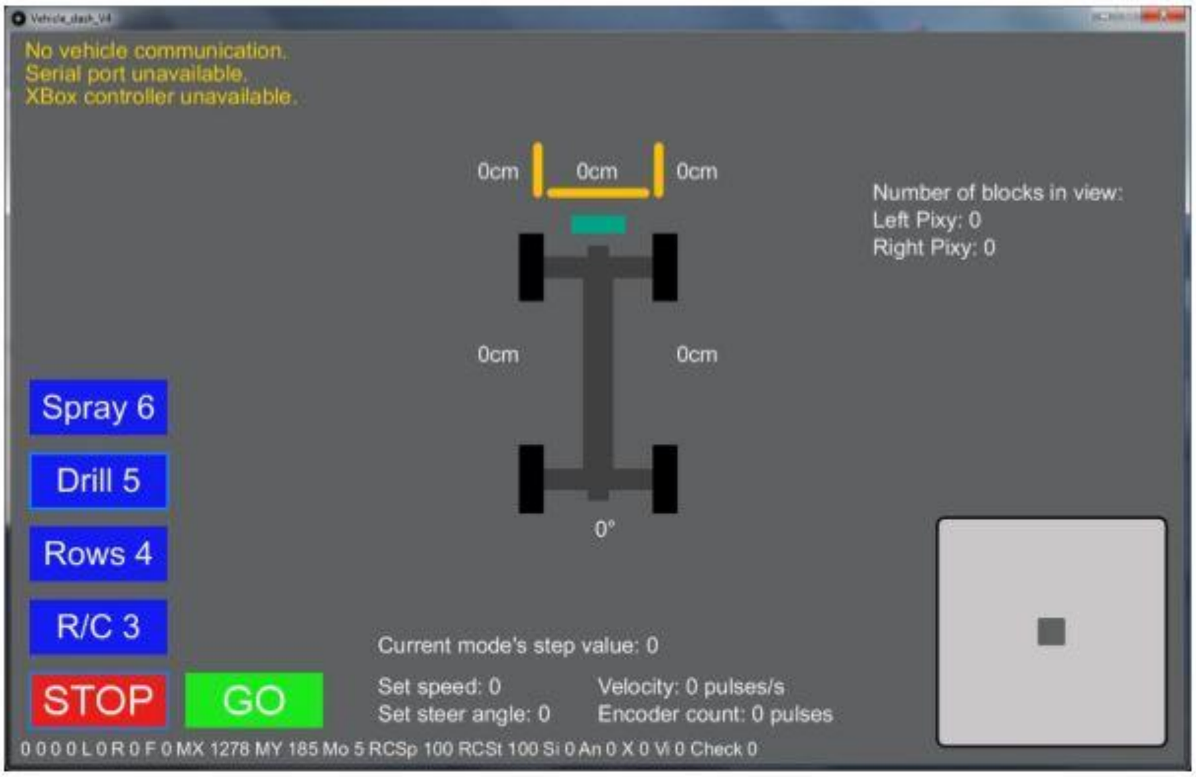
Figure 13 Arduino Mega ADK (L) and Arduino Micro (R)

The Mega ADK was selected based on its processing power, program memory size and number of serial ports (4). As it would be storing the program for all four tasks, a large memory was required. The serial ports, as seen in figure 3, were required in order to communicate with the HMI elements. Processing power was not as critical as first imagined as many time critical functions had been moved. Instead, it was important for the Micro. As the code was smaller the memory size required was not as great, and physical size took greater importance. Cost for both of these types of controllers was also one, if not two, orders of magnitude less in comparison to an embedded computer.

To achieve remote control, a wireless serial link was created between the robot and a laptop using a set of XBee Pro 10mW 2.4GHz RF modules. Matched with a 200mm aerial, these modules were chosen as they are capable of transmitting over a distance of 1.6km outdoors. Although over-specified for the current application, these will hopefully provide a resilient link for future applications. Through a USB-serial convertor board, data could then be exchanged with the robot. A Dell Latitude XT2 tablet/ laptop ran the HMI, allowing inputs via mouse, touch or an Xbox 360 handheld controller.

Human Machine Interface

As mentioned, the final part of the autonomous system was the user interface. Programmed in Processing 3, it received the serial data from the robot and displayed it for the user. Input via touch; mouse or an Xbox 360 controller was sanitised and then data was also sent to the robot, including stop/ start and mode commands.



No vehicle communication.
Serial port unavailable.
XBox controller unavailable.

Number of blocks in view:
Left Pixy: 0
Right Pixy: 0

Current mode's step value: 0

Set speed: 0
Set steer angle: 0

Velocity: 0 pulses/s
Encoder count: 0 pulses

```
0 0 0 0 L 0 R 0 F 0 MX 1278 MY 185 Mo 5 RCSp 100 RCSt 100 Si 0 An 0 X 0 Vi 0 Check 0
```

Figure 14 Human Machine Interface

The box in the lower right corner provided an alternative to the Xbox 360 controller for RC navigation, by touching and dragging the small grey square, the speed and steer angle could be controlled. The mode and start/ stop was controlled and indicated by the buttons in the lower left corner, whilst error messages were displayed in the upper right. The centre displays various data from the robot, including various sensor values. Finally, information such as that from the cameras (on the right) was only displayed when in the correct mode.

Performance & Recommendations

Following the robot performance during tests and competitions, various shortcomings were identified. For the vehicle itself, one of the major problems is the turning circle- three point turning. In the procedure of tests, the vehicle has an excellent performance, however, it cannot conduct turning in the competition field.

The changes from surroundings and terrain were supposed to be the main cause. Besides, human behaviour (visitors were too close at the turning point) may influent the detection of the turning point. Another issue is the battery voltage is unstable.

The end

The FRE 2017 is an unforgettable event during our study here in the UK, which gave us the opportunity to learn from other countries. And it gives us a profound lesson on teamwork. Although we are not awarded any prize, the experience is precious. We would like to thank all the 14 teams and especially the staff at Harper Adams University.

syngenta
CLAAS Stiftung
YARA
Knowledge grows
EurAgEng
Harper Adams University

The Great Cornholio

University of Applied Sciences Osnabrück

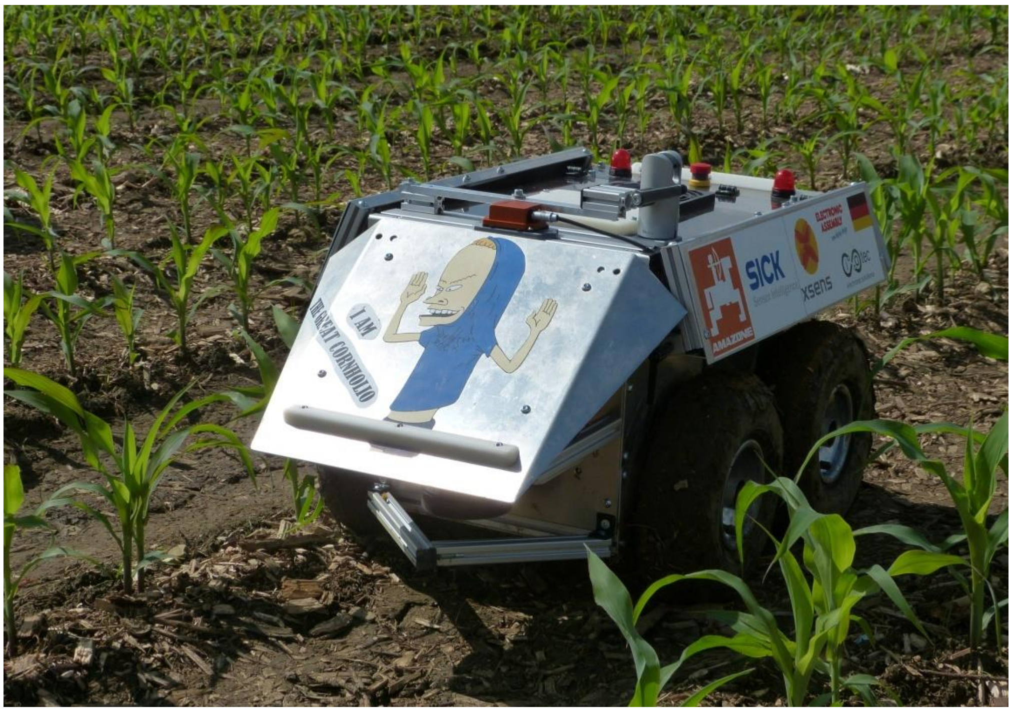


Figure 15: Field Robot, “The Great Cornholio”

Author names and Institution/Affiliation.

Matthias **Igelbrink**, Tristan **Igelbrink,** Steffen **Hellermann,** Jan **Roters,** Florian **Wasmuth,** Thomas **Ludemann,** Alexander **Kemeter,** Jannik **Redenius,** Jaron **Martinez,** Andreas **Linz,** Arno **Ruckelshausen.**

University of Applied Sciences Osnabrück
Engineering and Computer Sciences
Albrechtstr. 30
49076 Osnabrück Germany

Introduction

The following work is intended to describe the hardware and software used by students of the University of Applied Sciences Osnabrück for the 15th annual Field Robot Event held at Harper Adams University, Engineering Department, United Kingdom. The paper begins with a general mechanical overview of the referred to entry, “The Great Cornholio”, followed by a more in-depth description of hardware used, including specifications, and an overview of the software algorithms used to accomplish general functionality and the applicable tasks for the competition. A conclusion then follows to summarize the findings of the design process.

Mechanics

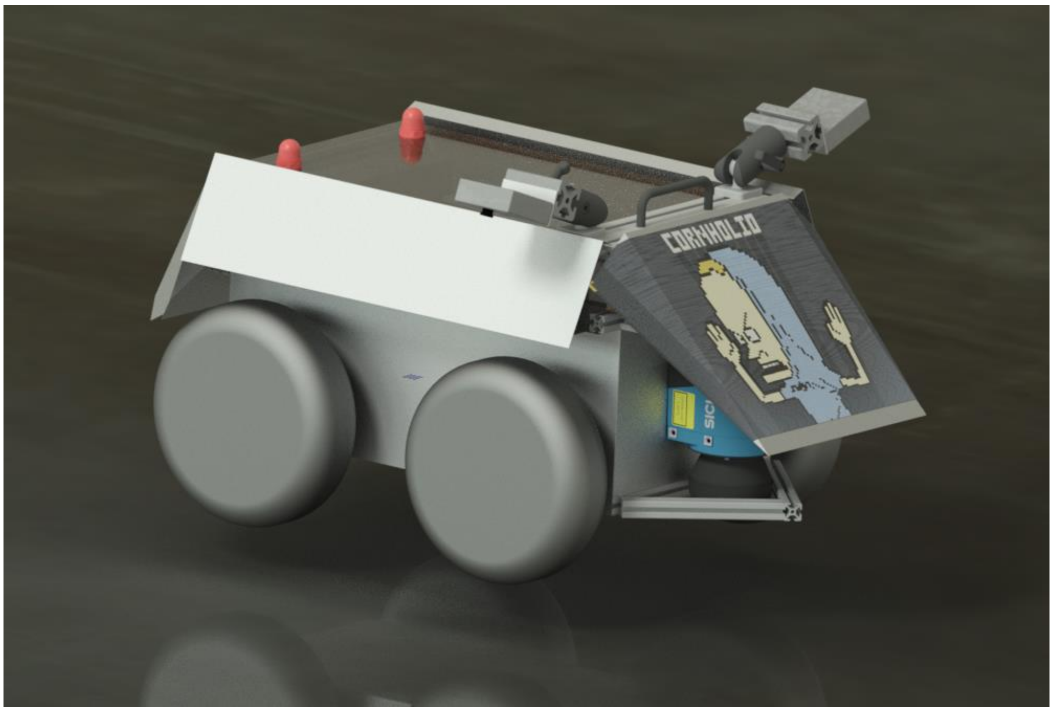


Figure 16: CAD draft of the Robot

"The Great Cornholio", whose CAD draft can be seen in figure 2, has an overall size of 54x69x44cm. As it is typical for robots based on "Volksbot" - the robot-platform of the Fraunhofer Institute – Cornholio's case relies on item profiles, making it easy to mount sensors, motors, etc. The profiles and the cover panels are made of aluminium. As a light metal, the usage of aluminium saves weight so the robot will be able to move faster. When the top plate is removed, two claps provide fast access to the fuse-boards and cables.

Two Maxon motors with an epicyclic gearing power the robot with a torque of 15Nm. One wheel on each side is connected to the motor shaft by a claw coupling. The other wheel is then connected to the first by a chain gear. Separating the drive sections makes it possible to control each side of the robot independent of the other as it is typical for skid drive. An advantage of skid steering is the low turn-radius, thus optimizing manoeuvrability. This behaviour is useful in navigating through the narrow curved rows of the contest environment.

As already mentioned, the robot's case consists of item profiles. Item offers a special ball joint fitting in their profile system that we used to mount our camera. It can be slided on a horizontal line using the clamp lever to change the camera's roll-pitch-yaw angle. This provides great flexibility for the image processing sensor's field-of-view. The camera can be shifted simply by loosening the lever providing flexibility and adaptive capabilities in a changing environment (e.g. from the laboratory to field conditions) within seconds.

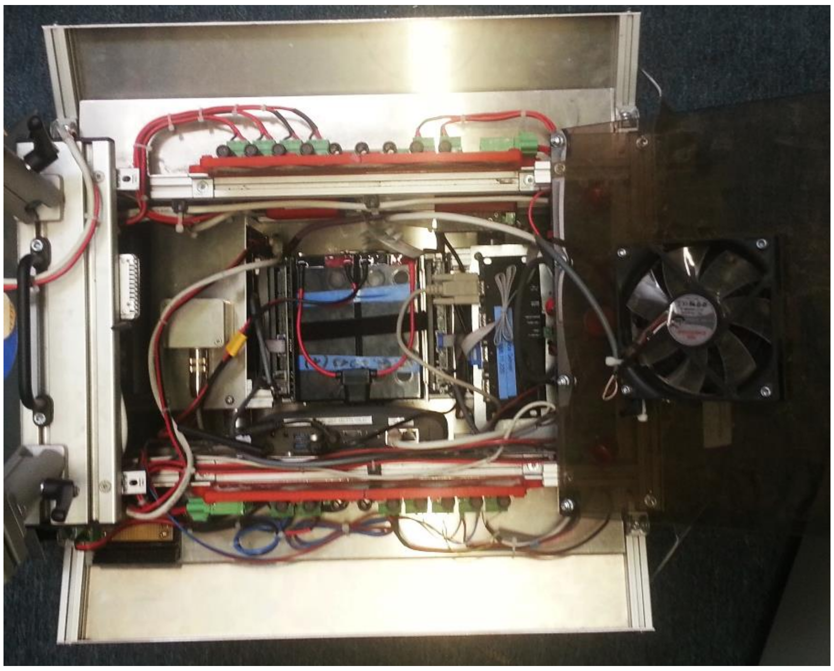


Figure 17: Inner assembly of the robot

Figure 3 shows the inner assembly of Cornholio. The robot is accessible through a PMMA lid, which is translucent. In the lid itself a PC-fan is embedded. The fan creates a positive pressure inside the robot to keep particulate material out of it.

The inside of the robot consist of the actuating elements, controls, a sensor and the power supply system. Right below the lid on the left and right side are the fuse-boards. Below the fan, the device server is fixed with adhesive hook-and-loop tape, which is connected to an aluminium plate. Just below the plate one of the motors is mounted. If there are any problems with the motor, the plate can be removed. Another plate is mounted over the second motor, which is used to fasten the IMU. The motor-controllers are positioned right next to the motors. The battery is stationed in the middle right between the two motor-controllers. They are held in place by aluminium brackets and plates which give a lateral bearing of the battery. This allows a fast access and replacement of the battery. The PC is placed on the left side right next to the battery. It is attached to an extra plate, which is mounted on the frame of the robot. The Ethernet-switches are placed in the front and rear of the robot. They are mounted in an upward position, making the status LEDs observable.

Power supply

The power supply is based on two 12V lead batteries running in series to provide 24 V DC. They are directly connected to a power supply board that provides three fuse secured voltage levels - 5V, 12V and 24V. The 5V and 12V voltage levels are realized by board intern step-down converters and have a maximum current of 5A. The 24V level is able to provide 20A without conversion. Additionally, there is a buck converter connected to the 24V port of the power supply board that provides additional current to the 12V output port.

Fuse Board

The fuse board connects all electrical components of the robot with their corresponding operating voltages and protects them from exceeding voltage specifications as well. The fuse board supports three different voltages - 5V, 12V and 24V. Every route is fused separately with easily accessible fuse holders for easy exchange.

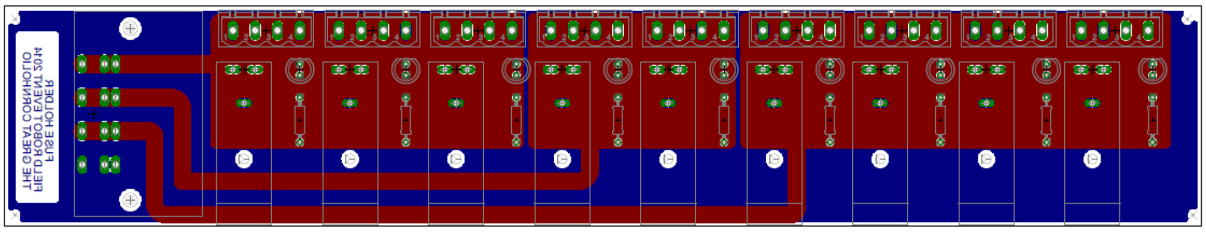


Figure 4: PCB-design of the fuse boards

The proper functionality of each fuse is displayed by a red LED. A working fuse bypasses the LED. The bottom side of the fuse board is moulded with epoxy resin. The advantage of this layer is the higher stability and contamination protection of the entire board. The electrical parts are connected with this board through a four-pin connector. This allows a hard-coded pin combination to prevent the interchange of operating voltages.

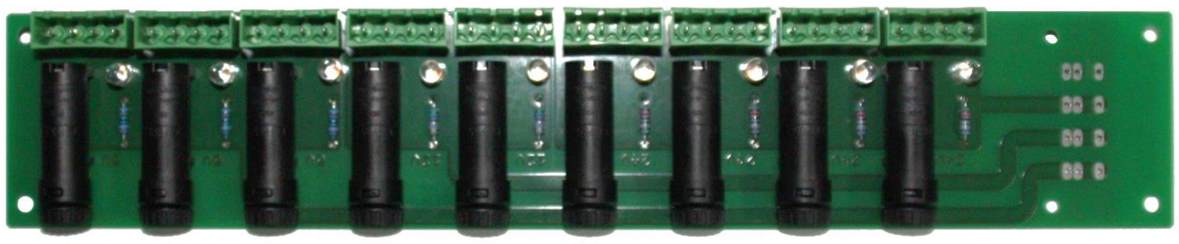


Figure 5: Fuse board

Computer:

- Model: Pokini-i
- Processor: Core i7-3517UE
- RAM: 8GB DDR3 SO-DIMM
- HDD: 120GB SSD
- Bluetooth integrated
- Passive cooling

Figure 6: Pokini i PC
Source: http://www.pokini.de/images/banners/pokini-i/1_pokini-i-composing.png

The computer is used to control the whole robot. It is connected to all devices by Ethernet.

Motor:

Model:	Maxon RE 40
Nominal voltage:	24 V
Nominal speed:	6930 rpm
Nominal torque:	170 mNm
Nominal current:	5.77 A
Stall torque:	2280 mNm
Stall current:	75.7 A
Max. efficiency:	91 %

Figure 7: Maxon motor
Source: http://www.maxonmotor.com/medias/sys_master/8797180919838/RE-40-40-GB-150W-2WE-mTerminal-Detail.jpg

Motor controller:

Model:	Maxon Motor EPOS2 70/10
supply voltage VCC:	11…70 VDC
Max. output voltage :	0.9 • VCC
Max. output current Imax (<1sec):	25 A
Continuous output current I_{cont}:	10 A
Switching frequency:	50 kHz
Max. efficiency:	94%
Max. speed:	100 000 rpm

Figure 8: Motor controller
Source http://www.maxonmotor.com/medias/sys_master/root/8797301768222/PRODUKTBILD-EPOS-2-70-10-375711-Detail.jpg

For the motor-control there are two integrated motor-controllers used to manage the speed and position of the wheels.

Display:

- Model: Electronic Assembly EA KIT129J-6LWTP
- Supply voltage: 5 VDC
- Resolution: 128x64
- Integrated touch panel
- Programmable with PC
- Connected by RS-232

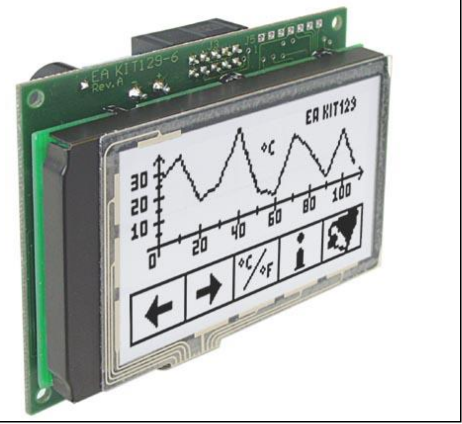


Figure 9: Touch display
Soure: http://www.lcd-module.de/pdf/grafik/kit129-6.pdf

The display with integrated touch panel generates a graphical user interface used to control the robot's functions.

Device server:

- Model: EX-6034
- Supply voltage: 5 VDC
- Chip-Set: Samsung S3C4510B
- Data transfer rate Serial: 50 Baud up to 115.2KBaud
- Data transfer rate Ethernet: 10/100Mbps

Figure 10: Device server
Source: http://www.exsys.de/media/images/org/EX-6034.jpg

The device server builds a connection for all serial-connected devices to the Ethernet.

Digital I/O converter:

- Model: EX-6011
- supply voltage: 5 VDC
- Digital Input Lines: 4, CMOS/TTL Compatible
- Digital Output Lines: 4, 2 non-inverted and 2 inverted
- Data transfer rate Ethernet: 10/100Mbps

Figure 11: Digital I/O converter
Sourcehttp://www.exsys.de/media/images/org/EX-6011.jpg

The digital I/O converter is used to switch loads controlled by Ethernet messages.

Software and strategy

In order to add modularity to our system, every hardware part of the robot communicates directly or indirectly over TCP/IP protocols. This set up allows to access, test and control the system or any part of it from any computer over a simple wireless router.

The control software:

The core of the software is based on the Robot Operating System (ROS) and makes use of the built-in state-machine functionality to perform the main control procedures. The software initially starts the state-machine server and decides which action is to be carried out depending on the present task and determination of the robot's state. Each action defines a set of different parameters and instructions to be carried out in order to reach a specified goal. Once the end of the action is reached, the action returns completion data and reverts control to the state-machine server. Examples of actions are row navigation, cease motor action and so on.

The navigation algorithm:

The employed navigation algorithm has proven reliable despite, or thanks to, its apparent simplicity. The navigation algorithm reduces the space in front of the robot into an occupancy matrix of four rows and four columns. The exact number and dimensions of the cells may change during operation to suit the current environment and task at hand. When the ranges of the laser scanner exceed a given threshold within a cell, the cell is labeled as unavailable and the robot will alter its course accordingly. In the headland we use our IMU sensor to perform a U-turn into the next row.

The basic moving algorithm for weed detection

The robot starts at the beginning of a specific row, depending on where the jury places the yellow golf balls. Now the robot starts moving into the row. Is there a yellow golf ball in the driving row, it will be detected and the robot drives backwards. If he was placed at the beginning of an empty row, the robot drives at the end of the row. How does the robot know in which direction it has to turn left or right at the headland of a row? It has an internal counter. Driving backwards to the start of the row increases the counter by 1 and driving forwards reaching the end of a row will increase the counter by 2. So the counter will ensure the robot to work through the field (depending of the starting position) from left to right or vice versa in one direction.

The basic moving algorithm for weed marking

The task`s requirement is to drive a specific path through the field from map data. The data has been created on the weed detection task. Unfortunately we were not able to connect the map`s data to a common path finding algorithm. So we created a state machine which works with coded moving commands from a table. The table could have been filled with commands from a path finding algorithm. We had the traveling salesman in our minds. Now back to the state machine. It performs all needed moving actions for the task. We filled the table manually and created an efficient path through the field driving over purple golfs balls and avoiding yellow ones.

Sensors /Actuators

Weed detection cameras

In order to recognize the purple and yellow golf balls in the field, we are using a metal rail, which is mounted on the front top of the robot. On the rail are three raspberry PI cameras installed. Each of them is looking into a specific row. The row on the left, the driving row and the row on the right are being continuously scanned of the task`s objective. The cams are connected to a raspberry pi which evaluates the incoming data from the all the cams. Once there is a hit on the objective, the PI sends a message via ROS node to the main computer to perform a moving action and making an acoustic signal for the jury.

Weed marking application

Before marking the purple golf ball it had to be recognized. The recognition works similar to the weed detection. Once a purple golf ball is detected on a back mounted Pi cam, a time delayed spray action is done. We use cans of football marking spray. They get pushed by small servomotors for releasing white and shiny foam.

Laserscanner

For the detection of plants and obstacles there is a SICK laser scanner LMS100 placed at the front of the robot. The SICK laser scanner uses the “time of flight” method for measuring the distance between the reflecting object and the scanner. This sensor is capable of measuring the surrounding area in an arc of 270° about its vertical center axis.

IMU

The robot uses the Xsens IMU (Inertial Measurement Unit) which incorporates accelerometers, gyros and magnetometers. The sensors are used to determine the speed and position/orientation of the robot.

Incremental encoder

The motion of the motor itself is measured with an encoder which is integrated into the motor. The information of the encoder and the information from the IMU are used to increase the accuracy of the positioning.

References

- Figure 6: Pokini i PC: http://www.pokini.de/images/banners/pokini-i/1_pokini-i-composing.png (Access: 11.07.2016)
- Figure 7: Maxon motor: http://www.maxonmotor.com/medias/sys_master/8797180919838/RE-40-40-GB-150W-2WE-mTerminal-Detail.jpg (Access: 11.07.2016)
- Figure 8: Motor controller: http://www.maxonmotor.com/medias/sys_master/root/8797301768222/PRODUKTBILD-EPOS-2-70-10-375711-Detail.jpg (Access: 11.07.2016)
- Figure 9: Touch display: http://www.lcd-module.de/pdf/grafik/kit129-6.pdf (Access: 11.07.2016)
- Figure 10: Device server: http://www.exsys.de/media/images/org/EX-6034.jpg (Access: 11.07.2016)
- Figure 11: Digital I/O converter: http://www.exsys.de/media/images/org/EX-6011.jpg (Access: 11.07.2016)

Team sponsors:

- AMAZONEN-Werke H. Dreyer GmbH & Co. KG
- Sick AG
- Xsens
- iotec GmbH
- Electronic Assembly GmbH

UniCorn

Janna Huuskonen[1)], Eljas Hyyrynen[1)], Jaakko Mattila[1)], Riikka Soitinaho[1)], Vili Väyrynen[1)], Tanvir Hossain[2)], Duc Pham[2)], Sampsa Ranta[2)], Aleksi Turunen[*)], Timo Oksanen[*)]

[1)] *Aalto University Department of Electrical Engineering and Automation*
[2)] *Aalto University Department of Mechanical Engineering*
[*)] *Instructor & Supervisor*

Abstract

The idea of developing field robotics is to alleviate the human workloads in the agricultural fields and making the work done in more efficient and accurate manner. Keeping this in mind, a group of students from Aalto University, Finland developed a field robot, named UniCorn, which took part in the Field Robot Event competition in the year 2017 held in United Kingdom. The Field Robot Event is an annual competition for promoting agricultural robotics [1]. This year the competition tasks were basic and advanced navigation, mapping, and precision spraying, while freestyle task was a standalone challenge.

UniCorn is an upgraded version of the previous year's field robot, however, the software has been developed from the very beginning. Several tools have been used in the design of the mechanical parts, mechatronics and software, consisting of CREO, Matlab, Simulink, Visual studio, KiCAD and CodeVisionAVR.

In the competition, UniCorn was able to secure first prize in advanced navigation task, and third prize in the basic navigation and the freestyle tasks. The robot did not start in tasks weed mapping and precision weed spraying.

Keywords: Field Robot Event 2017, field robotics, navigation, agricultural robotics

1 Introduction

The purpose of this project has been to develop an agricultural field robot, UniCorn, in respect to the instructors' requirement specification and the Field Robot Event (FRE) 2017 competition ruleset. FRE 2017 was the 15th in the series of the annual FRE competition, held this year at Harper-Adams University, Shropshire, United Kingdom. The tasks of the competition change yearly. This year's competition tasks were basic and advanced navigation, mapping, and precision spraying. Additionally, the freestyle task was and has been a standalone challenge in these competitions.

The robot UniCorn is the outcome of the effort of a group of eight students from Aalto University, of which three are from Mechanical Engineering and five from the Electrical Engineering and Automation department. The robot consists of previous years' robot parts, such the chassis and the wheels, and the new parts constructed this year, such as the whole software architecture. The new parts were constructed either for learning purpose, provide new features, or to improve the existing components.

Since none of the team members had the skill and knowledge of expertise level, some instruction and guidelines were followed using previous years' reports and instructors' expertise. Considering the huge scope of the field of agricultural robotics, internal cooperation was necessary to ensure the final goal.

The project started in the beginning of January 2017, about five months before the competition. The total length of the project was seven months. The project served as a foothold to the world of practical engineering with the challenge of a project team environment and excellent technical challenges to tackle.

2 Mechanics

This chapter discusses the different aspect of the robot's mechanical components, such as the mechanical design and manufacturing. Majority of the mechanical components were utilized from the previous year's robot [2, p. 5-11] while upgrading and expanding as required.

2.1 Axle Modules

The axle modules were used to control the steering and velocity of the robot's wheels. [2] The robot axle module design was inherited from previous year as is. For this design, it was a given as requirement for the project to work on the backslash seen in the drive train to improve the observability. The shaft and universal joints coupling was considered and given as objective to improve. The design was revised and new parts were fabricated to address the issue:

- Reducing axle backslash by improving positioning tolerances and clearances at axle universal joint and differential
- Previous design used M4 DIN 913 set screw, design was with threads in universal joint and flat bottom hole
- First improvement iteration M3 DIN 912 8.8 screw with threads in axle and 5 millimeter hole in universal joints, screw head was brought from 5.5 to 4.95 millimeters with more precision
- First iteration with 8.8 screws had severe failures as multiple screws were found broken by shearing while the robot was being tested for the tasks. The previous design used M4 screws and these did not have similar issues. As an resolution to the issue, 8.8 steel screws were replaced with 12.9 steel screws to have more allowance on forces.

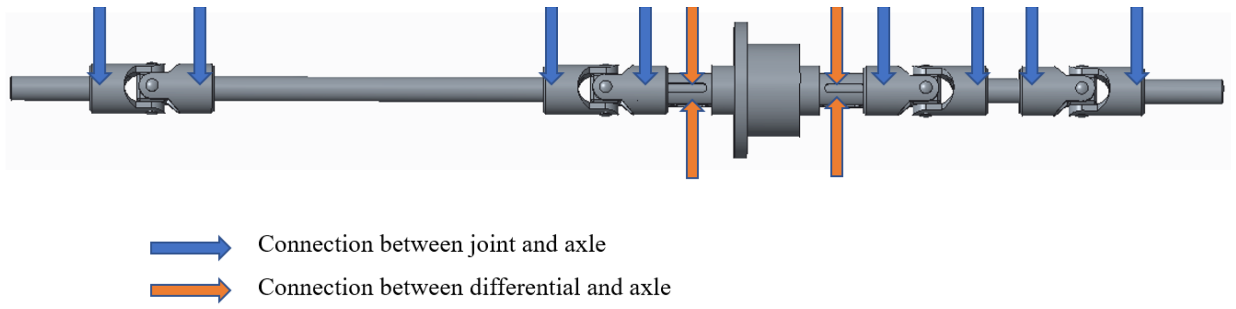

Connection between joint and axle
Connection between differential and axle

Figure 1: Axle module shaft.

As represented in the figure above, there are a total of twelve M3 DIN 912 12.9 screws which were used in this improvement. Out of these screws, the head of four screws used in the connection between the differential and the axle were turned to be tight clearance fit with the slot on the differential, close to 4.15 millimeters. Moreover, those header screws also had to be lowered by 0.5 millimeters, so that the axle could move more easily and not be blocked when assemble with different mechanical parts in axle module. Besides that one differential-axle connection requires two screws, hence, the length of those screws also needs to be reduced to 5 millimeters. The detailed dimensions of M3 screws is presented in [3] and [4].

2.2 Local UI mounting

Beside the autonomous mode, the robot is also equipped with a physical user interface (UI), which allows the user to control the robot without a remote controller and display important system status information. Some functions of user interface were required by the rules of the competition, so some buttons and indicators were mandatory. The local UI (LUI) consists of LEDs, functional buttons and LCD which will be discussed more in details in the Hardware section. All the components of LUI need to be properly mounted, as well as in the logical arrangement to provide an easy accessing and monitoring to the user. The buttons were provided by Valtra company, and they are the same used in tractor cabin which need to be mounted with hexagonal nuts, capable of tolerating harsh conditions. Besides that, there was several issues which lead to redesign the Local UI mounting part:

- The previous designed was attached directly to the hard plastic plate with screw and nut underneath, so it was very difficult to assembly and disassembly. Moreover it also created the crack on the plastic plate because of tightening.
- The buttons and LEDs mounting position on the previous design was unsymmetrical, and the annotation space was also quite small.
- The change of hardware design because instead of using one PCB as previous team, we intended to make the LEDs PCB and the main Local UI PCB separately, so the mounting places were also different.

Based on that, the new mounting place for local UI was designed as shown in the figure below. The shape of the design is based on the previous year version to fit the cover design, but the position of the buttons, LEDs and LCD were changed so that it would fit the PCBs boards. The new version of LUI mounting part was attached to the mast by three bolts to increase the stability, reduce amount of unnecessary nuts and screws, also make it more convenient to assemble and disassemble. Using CAD sheet metal model design, the LUI mounting part was cut out of a two millimeter thick aluminum sheet with laser cutting and then bent to reach the desired shape. By the way, with this new design, the LUI PCBs boards could also be mounted and taken out easily for programing, testing and fixing.

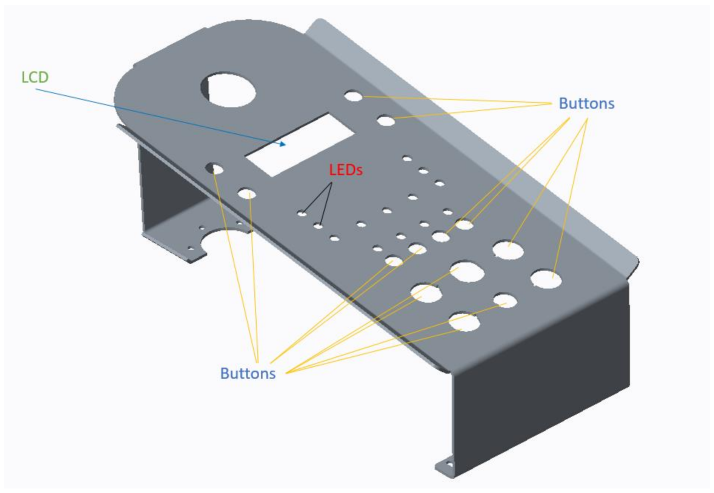


Figure 2: LUI mounting part.

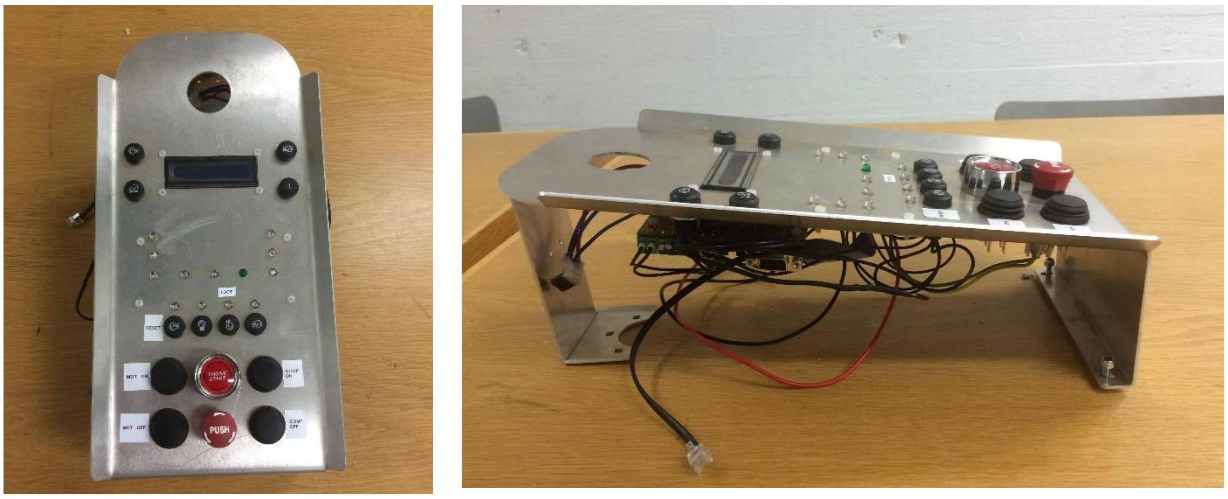


Figure 3: LUI after manufacturing.

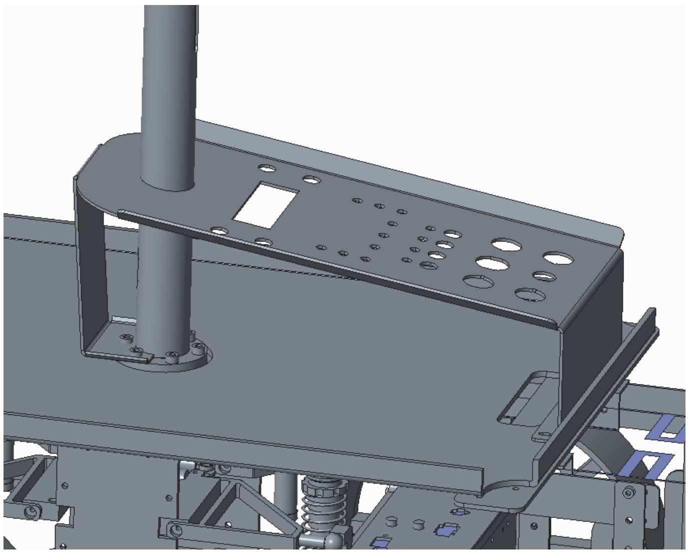


Figure 4: LUI assembly.

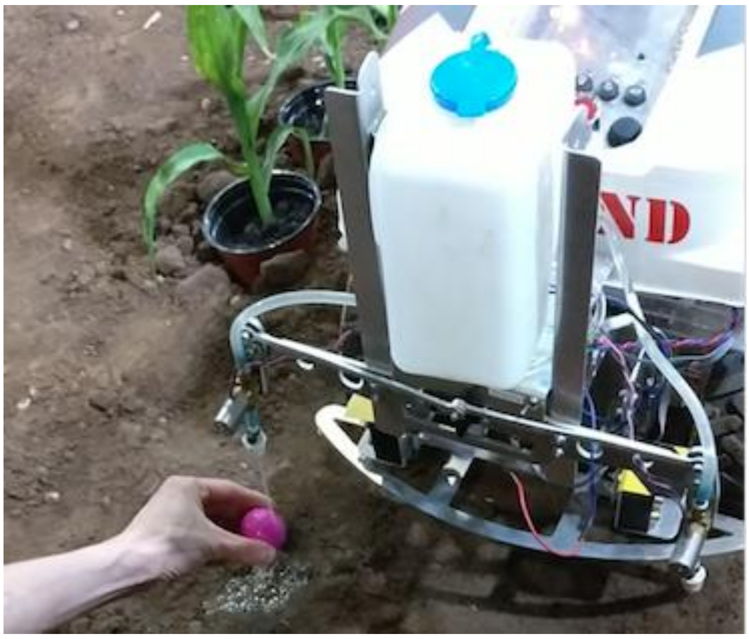
2.3 Sprayer

Figure 5: Testing the sprayer implement at FRE event, ball effectively gets wet.

The competition task 4, precision spraying, called for a method to spray water on balls, representing weeds, with some accuracy, hence, a sprayer was required. This implement sprayer design is also mostly inherited as design from previous year course robots. The sprayer was first introduced in Carnivore [5] and design was revised and improved for Agronaut [2]. The implement hardware for Unicorn is based on these with modifications. This base hardware consisted of a water tank, a pump, a guidance servo to enable the sideways movement and two solenoids driving two separate sprayer nozzles. The attachment and some steel bars in the construction were revised and replaced with more lightweight aluminum parts.

With proper calibration, it was possible to target one of the nozzles to a ball on the track within the precision constraints given by the task. The pump was powerful enough, hence, the required pressure level was reached rapidly to shoot water stream through the nozzle. Figure 5 above illustrates the working principle of the sprayer on target ball.

To calibrate the sprayer, a simple chart was made, consisting of x-position of the water stream



target with a given servo command given, together with nozzle selected. It was noted that the implement together with the water tank significantly change the robot overall weight distribution, and due to this, for example the front laser scanner tilts higher than without the implement potentially affecting the navigation.

2.4 Maintenance stand

The previous maintenance stand had some issues:

1. The height of the carrier was not good enough while inspecting the robot.
2. The robot is not balanced nicely on the carrier that gives movement in the sideways.
3. Clearance from the ground was low.
4. The stand was too big for the travelling luggage.

To address these problems, a new maintenance stand was designed and manufactured in order to better support the testing, the checking, the assembly and the disassembly processes. The proposed design for the stand is shown as in Figure 6 below, emphasizing the ease of disassembly and manufacturing. Since the robot was to be air transported, so the idea of new design was that the stand could be disassemble to small parts so that it can fit the travelling luggage. Therefore, the stand consists of sheet aluminium parts which were manufactured by combination of laser cutting and bending process, and very easy to assemble also disassemble by M4 nuts and screws. The two stands are placed under the two axle module using the four hole on top to make the tight fit connection with four headers of screws underneath the axle module, as shown in Figure 7. Furthermore, the side of the stand attaches to the side of the axle module which helps increasing the stability and the strength of the connection. After manufacturing, the two stands fulfilled the requirements by holding the robot stable while testing high speed driving. Based on the user experience, one more stand should be introduced in the future which could be placed under the middle chassis of the robot to conveniently support the axle module replacing task.

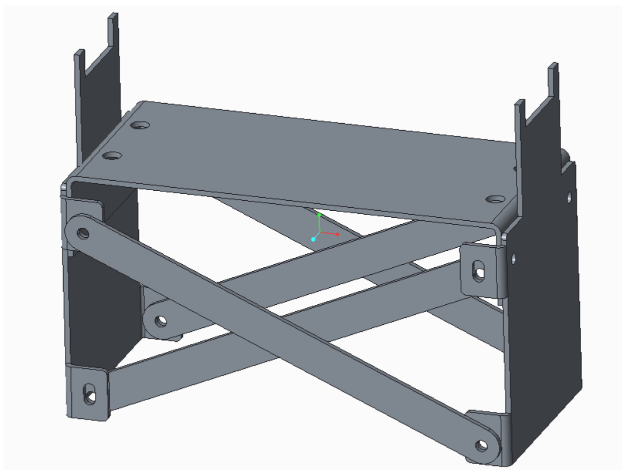


Figure 6: The maintenance stand.

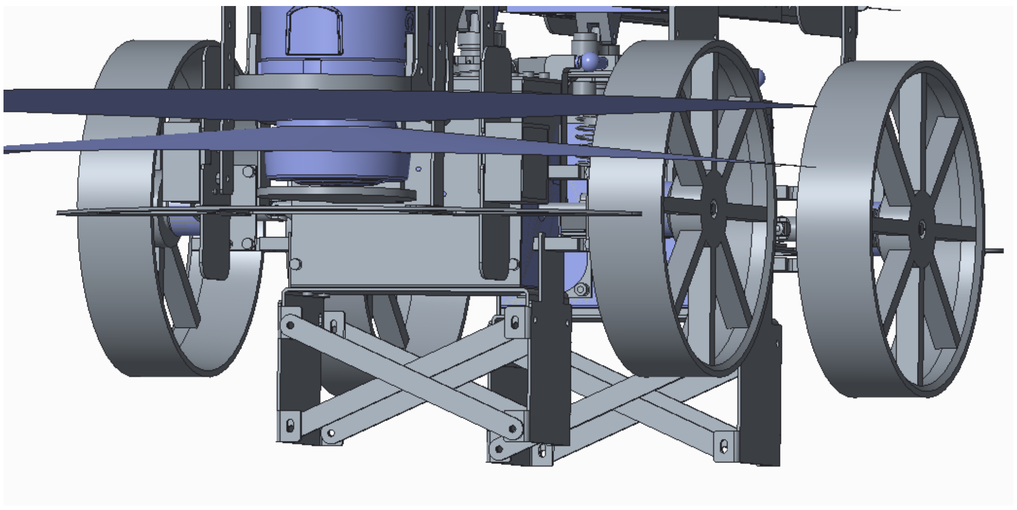


Figure 7: Robot's maintenance stands presented in CAD model.

2.5 Three cameras mechanical system

The task 3, mapping the field, required the robot to detect objects in the field within a brief amount of time. To spare time, a design was created for using three cameras simultaneously to scan several maize rows at once.

These three cameras required mounting which had to be designed. The requirements for this design were fixed as:

- Three cameras have to stand at the same height, and in straight line.
- The distance between two camera need to equal to the row width 75cm, so that three cameras can

see in three row in sequence.
Based on those requirements, the mounting design uses two 80cm long and two millimeters thick T-bars for mounting two side cameras to the existing mast of the robot. The middle camera was mounted on a Dynamixel AX-12A servo on the top of the mast using the mechanical part support for turning backward as demonstrated in Figure 8 below.

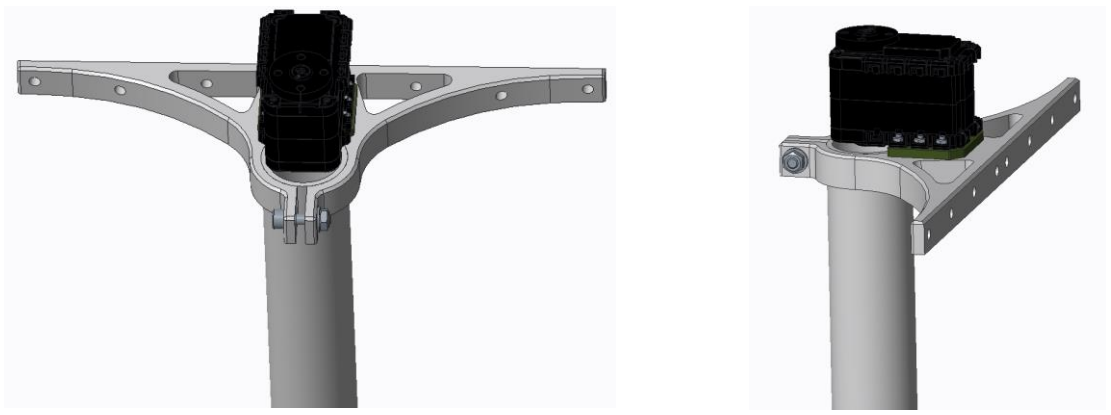


Figure 8: Middle camera mounted part.

This mechanical part was manufactured by CNC milling machine, and used to mount two side T-bar. The material for manufacturing of this part is aluminum, similar to the other metallic parts of the robot's design. This part's design focuses on minimizing weight (98 g) to aid the robot in maintaining stability when fully assembled.

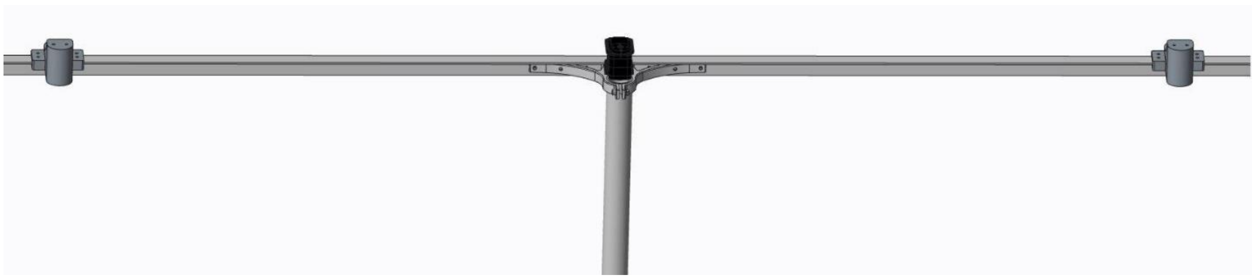


Figure 9: Three cameras mechanical system.

To make the three camera stand at the same height, the two side cameras also need to be mounted on the support parts which have a height equal to the height of the Dynamixel AX-12A servo. Presented in Figure 10, these support parts were made by plastic with 3D printing manufacturing for lowering the weight, and hence, maintaining the stability of the structure under sideways motion.

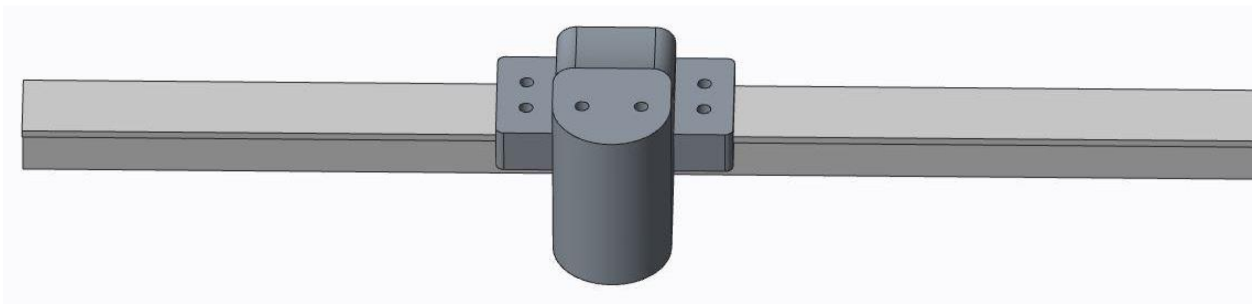


Figure 10: Side camera mounted part.

2.6 Cover

The cover was made to protect the robot, mostly the electronics from rain and dirt. It also made the appearance of the robot cleaner and served as a surface for logos and texts. The cover was made out of vacuum formed plastic. The mold was already designed and manufactured by GroundBreaker team (2015) [11]. Based on the raw cover-mold, the cover was shaped so that it can fit the robot and also the new local user interface as shown in the two figures below. Then, the cover was painted with a ‘red-black arrow' pattern, and decorated with the robot name, and the sponsor stickers.

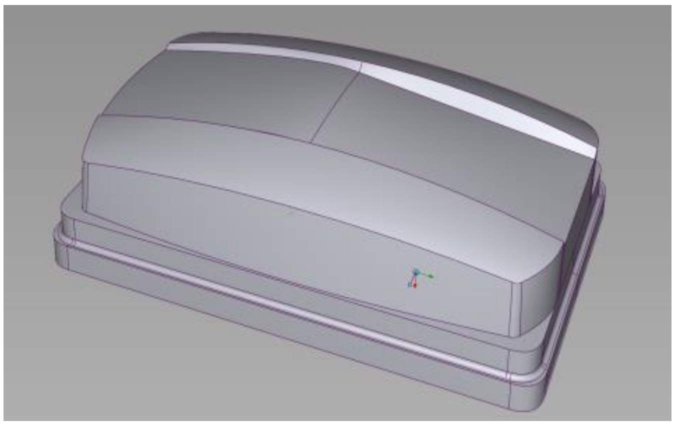


Figure 11: Cover mold.

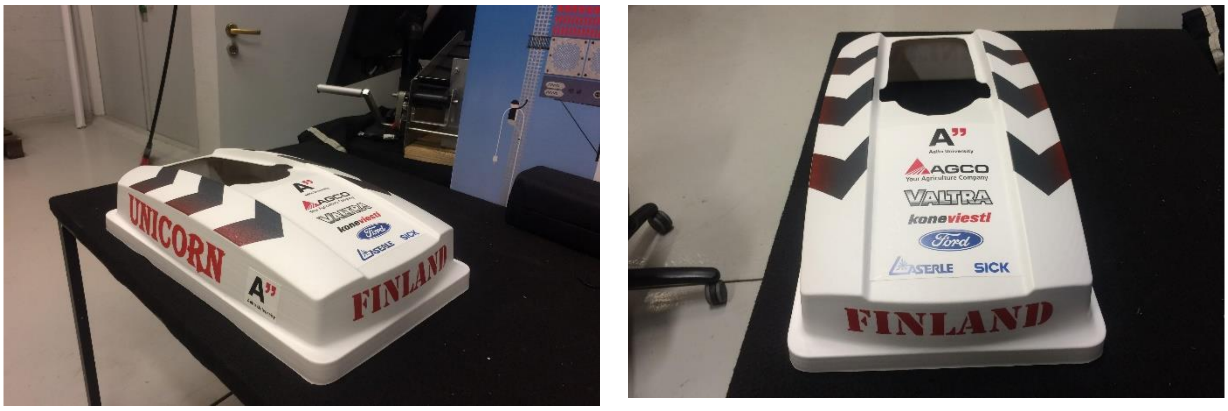
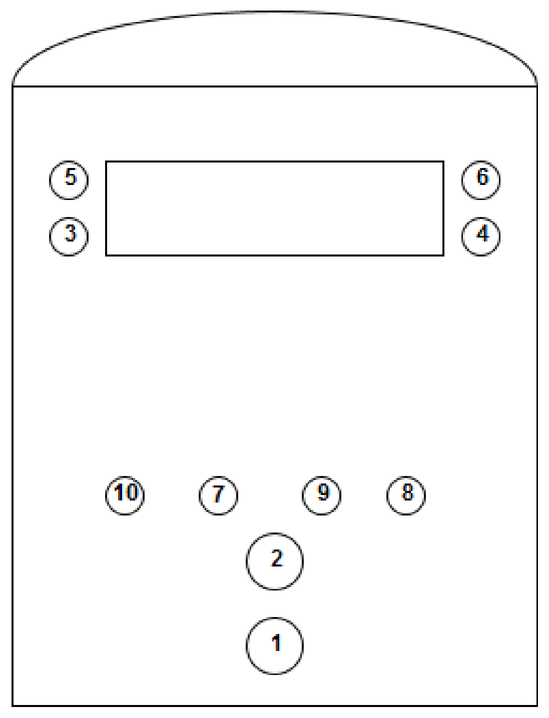


Figure 12: Cover after cutting and painting.

3 Hardware

3.1 Local UI Printed Circuit Boards

The LUI printed board was designed so that it can be easily assemble and disassemble with the mechanical LUI as mentioned in chapter 3. The idea was that there would be two separate PCBs, one was the main LUI, and one is LEDs printed board. They will connect to each other via the 16 pins connector. The main LUI consists of the Chip45’s Crumb 128-CAN V5.0 AVR CAN Module microcontroller [13] (Can Crumb), LCD, the connectors for 10 buttons, horn, beeper, mast LEDs, top camera servo and also the 6 pins connectors to the Sprayer board. The purposes of buttons and LEDs are represented in detail in Figure 13 and Table 1 below.



LUI buttons	Functionality
1	Stop
2	Start
3	LCD State shuffle previous
4	LCD State shuffle next
5	Previous mode
6	Next mode
7	Enter mode
8	Spraying toggle
9	Beeper toggle
10	Reset

Figure 13: LUI buttons mapping.

LED	**Color**	**Functionality (indication)**	**Enum**
motor CO	**AMBER**	Motor battery cut-off status	1
(unnamed)	**WHITE**	(none yet)	2
ALV	**BLUE**	CAN message alive indication (blink)	3
L TURN	**RED**	Next turn left (blink), currently turning left (static)	4
INROW	**BLUE**	Robot is currently in inrow mode	5
R TURN	**GREEN**	Next turn right (blink), currently turning right (static)	6
REV	**WHITE**	Robot is currently reversing	7
eBox CO	**AMBER**	eBox battery cut-off status	8
NUC CO	**AMBER**	NUC battery cut-off status	9
(button1)	**BLUE**	Corresponding button indicator, e.g. mode	10
(button2)	**BLUE**	Corresponding button indicator, e.g. mode	11
(button3)	**BLUE**	Corresponding button indicator, e.g. mode	12
(button4)	**BLUE**	Corresponding button indicator, e.g. mode	13

Table 1: LEDs indications and colors.

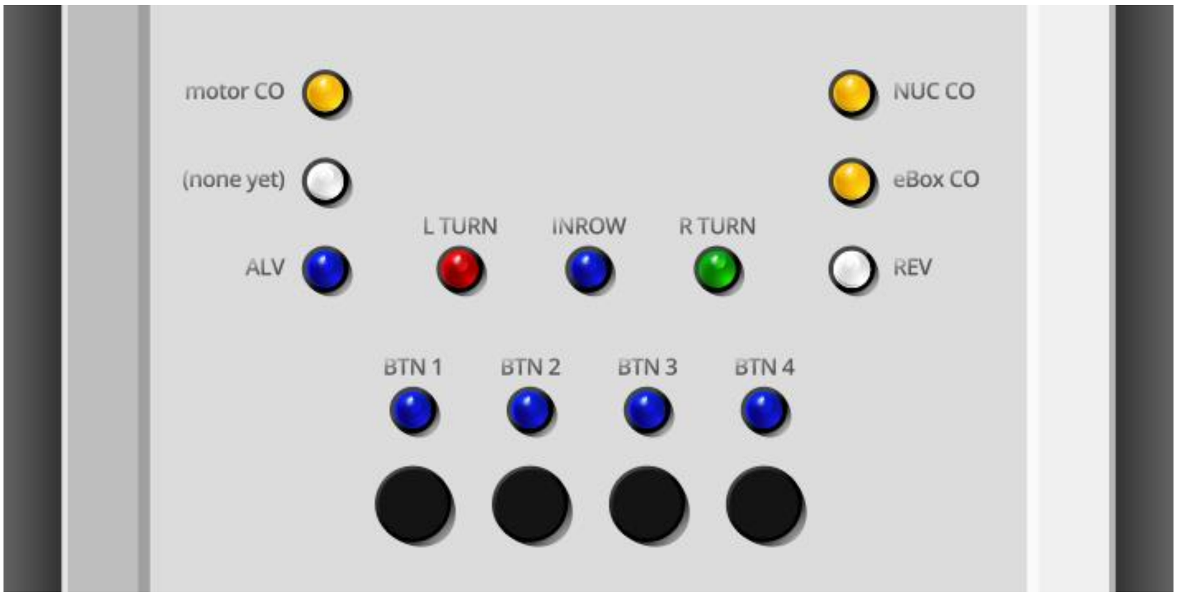


Figure 14: LEDs indications and colors.

Besides the LEDs on the printed board, there are three differently colored LED strips in the mast. These LED strips give information about the direction of the robot's movement. The upper LED strip is blue, turn right LED; in the center is green row navigation LED strip and lower LED strip is red turn left LED strip. The strips are used because of 360-degree visibility. However, the information to these strips and the horn should go via the Crumb which uses 5V voltage, while the strips and the horn use 12V operating voltage. This problem is solved by using MOSFET transistors as shown in Figure 15.

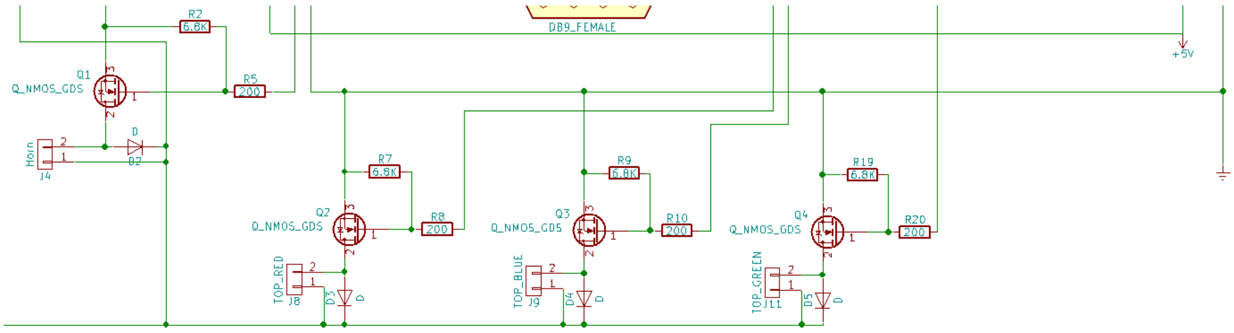


Figure 15: MOSFET using for mast LEDs and horn.

By using an Atmega 90CAN128 - microcontroller, the LUI communicates with robot system via CAN bus. As shown in the schematic drawing in Figure 16, the electrical board gets 12V voltage via J2 coupling from the CAN-bus. The Crumb use 5.0V operating voltage, for this reason regulator (7805) is used to decrease the voltage to 5V.

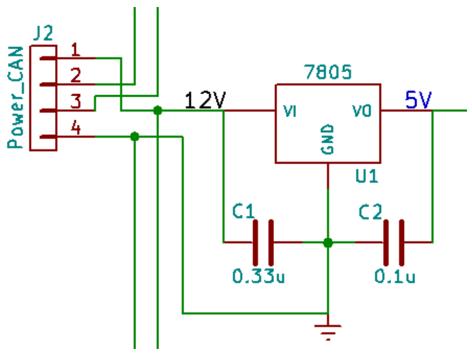


Figure 16: Power supply for LUI main board.

The top camera servo is a Dynamixel AX-12A robot actuator which was connected to the Crumb via serial port 1. It uses a 12V power supply and TTL Half Duplex Async Serial protocol [12].

Figure 17: Dynamixel AX-12A servo.

One of the design goals was to minimize the extra hardware components compared to the previous year’s robot. As an example of this, the LUI also be used to control the spray mechanism in order to utilize the Crumb’s pins, which removed one microcontroller from the design compared the previous year. As shown in the schematic drawing in Figure 18, there was a 6 pins connector which was used to connect with the spray PCB (will be discussed more in detail below). In which, two pins were used for power supply, and three pins to toggle two solenoids and pump motor. The last pin was connected with OC1A pin of the Crumb for servo PWM controlling.

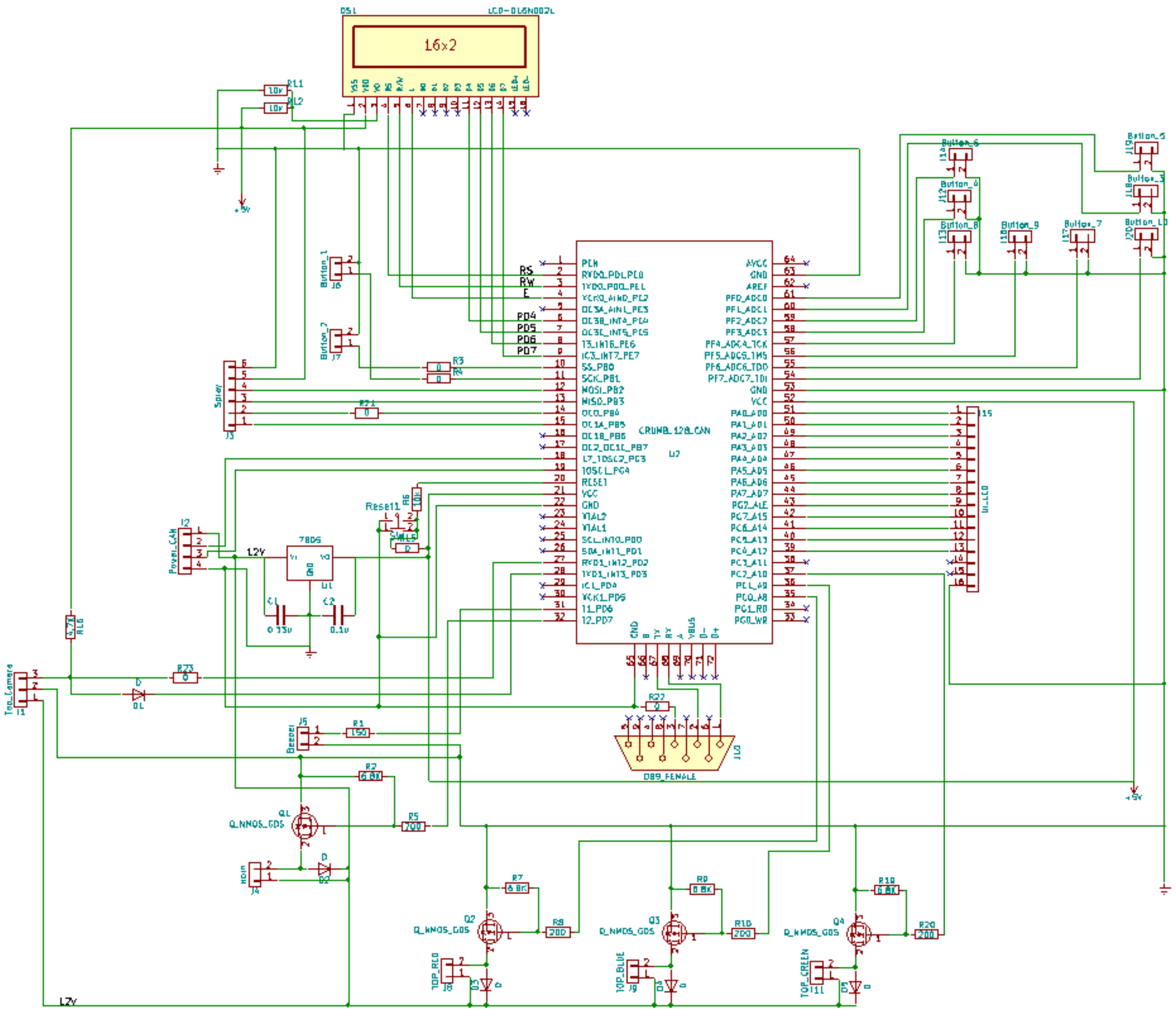


Figure 18: LUI main board schematic drawing.

3.2 Sprayer Printed Circuit Board

In accordance to the requirements of the competition task 4 rules, UniCorn was equipped with a sprayer. Based on that purpose, the sprayer mechanical mechanism was decided to derive from the last year robot. However, based on the change that the local user interface will control also sprayer, the new PCB board was designed for the sprayer.

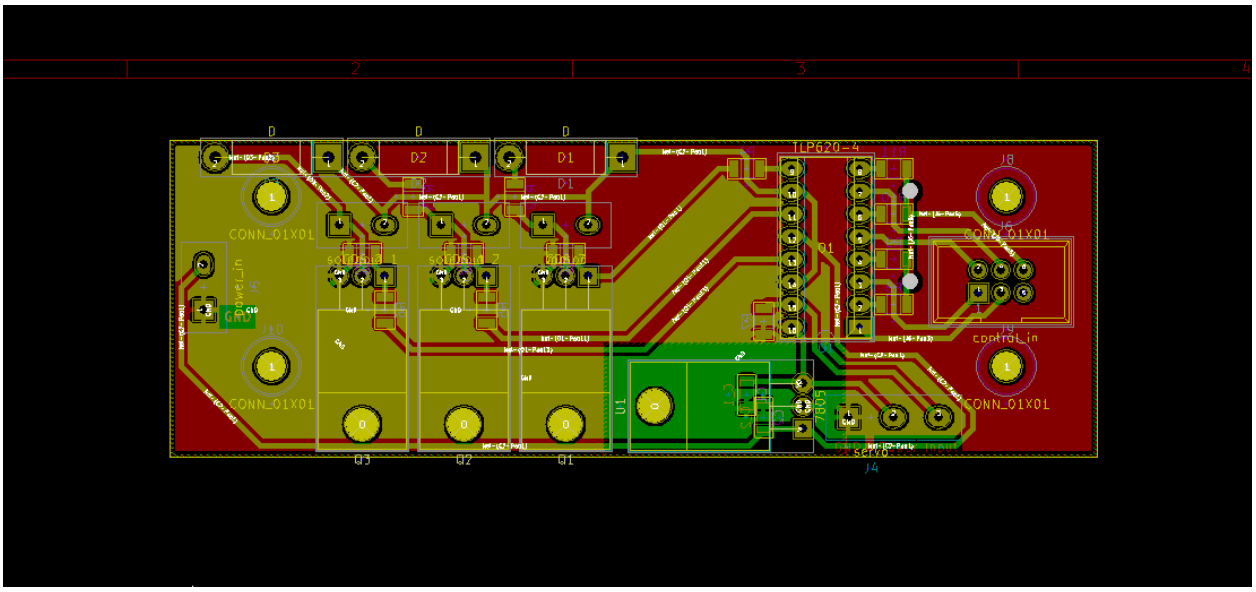


Figure 19: Sprayer PCB.

The sprayer board used the 12V power supply from the robot motor battery, and connected with the LUI using the 6 pins connector. Moreover, because two solenoids and the pump motor use 12V power supply, so there are three MOSFET were used here as switches to control them. Besides that, the TOSHIBA TLP620-4(GB) transistor output optocoupler was also used to interface the high voltage to the low voltage parts of the printed circuit board, thus keeping the computer power clean from the motor interference, through separating that and motor power completely. The IC 7805 was also used here to generate the 5V power supply for optocoupler and servo motor.

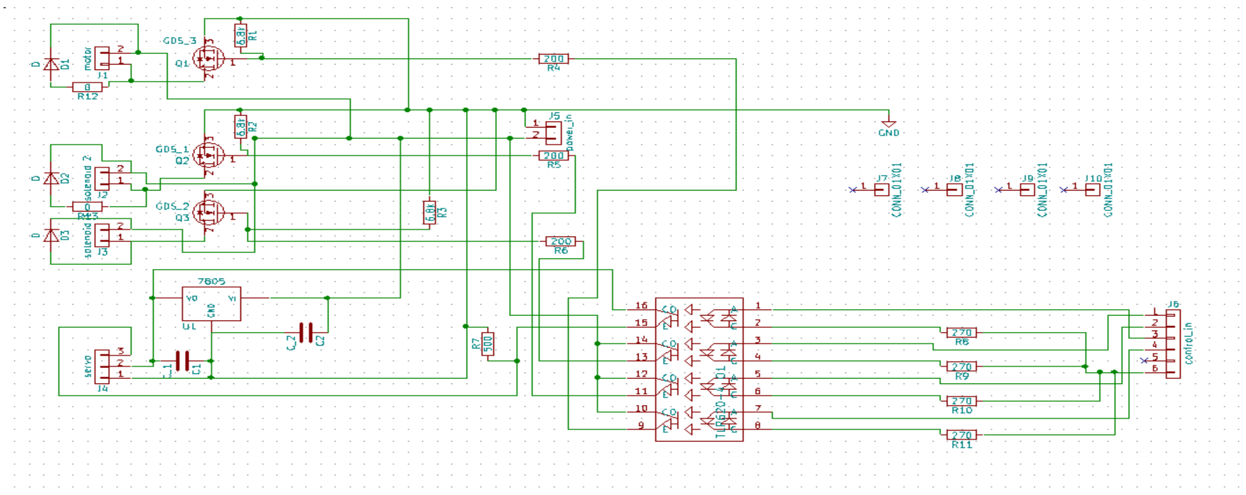


Figure 20: Schematic diagram of the Sprayer circuit.

The components used for the whole sprayer circuits are listed in the table below.

Type	Quantity	Library
Capacitor (C_1206)	2	KiCAD

Diodes	3	KiCAD
2 pin connector	4	KiCAD
3 pin connector	1	KiCAD
IDC connector (6 pin)	1	KiCAD
TLP 620-4	1	KiCAD
Resistors (6.8k, 200, 270, 500)	11	KiCAD
U1 (7805)	1	Own library
Mounting hole (3.2 mm)	4	KiCAD

Table 2: Components of the sprayer circuit.

3.3 PCs

The robot had two computers on board: eBox-3350MX and Intel's NUC with a Core i5 processor. Out of the two PCs, NUC had more processing power, which influenced the decisions regarding dividing the software between the two computers. Hence, eBox was responsible for receiving sensor data from RangePacks, gyroscope, and cut-offs. Then, these sensor reading were wrapped to CAN messages and sent for NUC to read. NUC was used for reading the rest of the sensors, reading and sending CAN messages, and operating computationally demanding tasks.

3.4 Sensors and drivers

3.4.1 Joystick

A Logitech controller was used for manual control of the robot as well as starting and stopping the robot in autonomous mode. The controller was connected to the NUC computer with USB.

3.4.2 RangePacks

The robot has four RangePack units, two on each side of the robot. These RangePacks were first implemented on the field robot Cornivore [5, p. 23]. Each RangePack unit has an ultrasound (SRF04) and infrared sensor (GP2D120). An ATMEGA88 microcontroller handles the measurements and communication via RS-485. These RangePacks were connected to eBox PC with custom made RS-485 communication hubs and a RS-485-USB-adapter. The data from the RangePacks was packaged according to a custom message protocol.

The ultrasound distance measurements were used in the algorithms developed in Simulink, discussed in chapter 4. The measurement range is between 0-2085 millimeters. However, a more limited range was used in order to rule out targets that were not desired maize rows. The infrared parts of the RangePacks were not utilized at all, due to their inferiority in the size of the sensory area.

Unlike the other device drivers, the driver for reading the RangePacks was provided to the team ready.

Finally, the RangePack information is postponed to the CAN bus.

3.4.3 Laser scanner

The robot was equipped with a SICK LMS100 2D laser scanner [6]. This laser scanner has a resolution 0.5 degrees, a scanning area of 270 degrees and a configurable measurement frequency of 25 or 50 Hz. In this application, a frequency of 25 Hz was used. The range of the scans is from 50 centimeters to 20 meters. However, in practice, the minimum range was smaller than 50 centimeters and the maximum range for this application was 2 meters. The operation environment of the robot was assumed to have uneven ground. As a result, laser beams with a longer range would reflect from the ground, not from the maize rows, which were the target of detection.

The laser scanner was connected to an Ethernet router and communication with the scanner was done with TCP according to a message protocol defined by SICK. One message from the scanner includes various data, such as the contamination status of the scanner and the distance measurements. One scan includes 541 measurement points for a scanning area of 270 degrees. The laser scanner was used in advanced positioning, row end detection and navigation algorithms discussed in chapter 4.

3.4.4 Gyroscope

The gyroscope unit originates from the field robot Cornivore [5, p. 24], where two VTI scc1300 D02 chips were used to comprise an inertial measurement unit with a two-axis gyroscope and a three-axis accelerometer. The chips were mounted such that they measure the robot’s roll and pitch axis angular velocities. A third VTI chip was later added for the field robot DoubleTrouble [7] to measure yaw angle. This three-axis gyroscope assembly was also in use for the field robot UniCorn.

The gyroscope was used for measuring the heading of the robot with regards to the original heading at startup, when the gyroscope is calibrated. The gyroscope transmits data using the same custom message protocol as the RangePacks. A state machine implementation was coded for parsing the messages. In addition to the device identifier, a message includes 16 bit values for roll, pitch and yaw angles.

3.4.5 Cut-offs

The cut-offs are used for measuring the voltage and current of the LiPo batteries. Moreover, the cut-offs monitor the voltage of the batteries and cut off power if the voltage of the batteries drops too low. This is done to protect the batteries from damage caused by exposure to too low voltages. The LEDs indicate if the voltage dropped under warning limits.

These cut-offs had been custom-made by the instructor for the field robot Agronaut [2, p. 18]. Like the RangePacks, the cut-offs are connected to a custom RS-485 hub and a RS-485-USB-adapter. The data from the cut-offs is transmitted using the ‘field robot protocol’.

3.4.6 Cameras

Three identical Microsoft LifeCam web cameras are used in the final robot setup. Initially, only one web camera was assembled with the robot mast and the direction was controlled with a servo. Due to the time constraints of the field mapping task, the robot has been designed to use a three camera setup to simultaneously to detect weeds and obstacles also on the adjacent rows from the robot’s current row.

To realize this setup, the cameras needed to be attached, calibrated, and set up so that their function would be rather identical, hence, lessening the need for separate tuning for each camera. An USB hub is used to extend the available USB ports of the NUC PC, however, even then only certain configuration of USB ports functioned with simultaneously plugging in the three cameras. More on the calibration will be found on the Machine Vision section of this report.

3.4.7 GPS

A Garmin GPS 19x NMEA 2000 module is available, but it was not used. The main reason was that using GPS was prohibited in the competition. Even if it would have been allowed, the competition was held indoors so it could not have been used. Therefore, writing a driver for the GPS was not prioritized and the time was used for developing more important parts. GPS data could have been a nice addition for the log files provided by the field testing.

4 Software

4.1 PC Control

All programs requiring more processing power are implemented on the computer NUC, whereas the computer eBox only hosted drivers for the cut-offs, RangePacks and the gyroscope. The sensor data from eBox was sent to a main program on NUC via CAN. The main program on NUC hosted all the primary functionalities of the robot, including the navigation and positioning algorithms done in Matlab Simulink. In addition, the machine vision program was running on NUC. The eBox program was written in C# using Microsoft Visual Studio 2008. The C# side of the main program on NUC as well as the machine vision program in C++ were developed using Microsoft Visual Studio 2013.

4.2 Toolchain

For the software running on NUC, the familiar toolchain from previous years was used. The core functionality and algorithms were developed in Matlab Simulink and code generated as C-code. A C++ dynamic-link library was then created from the C-code. The .dll was used by a C# program with platform invoke. Functionalities such as communication, log writing and device drivers were written in C#.

There were several advantages using Simulink in the toolchain. The atoms of a Simulink diagram are blocks with inputs and outputs. Blocks are connected to other blocks with wires and they can also contain other blocks inside them. A block can also be a piece of Matlab code which was very profitable and used feature in the software done for UniCorn. This environment provided a visual developer tool that illustrated hierarchy and interfaces intuitively.

The availability to draw things on the screen was used to make a simulator. This functionality was also used to plot graphs of the gathered log files. Thus, not only the logic but also a high-level analysis whether the logic works could be made in the same environment. There was also a tool called Stateflow, which is basically a block that holds a state machine chart inside it. The main architecture and several algorithms were built on this functionality. As problems could be solved with several ways, there were often many workarounds for the problems that were faced.

For the downsides, Simulink is not open source. It has limited the size of the community and answers to the numerous problems one faces during developing. The error messages often did not point to the real problem so the user had to stay on track with the errors by being strategic with the development process.

Another downside was the complexity of the toolchain. Compiling the entire program required several steps and numerous settings. In addition, the code generated input, output and tunable parameter structs that had to be manually updated each time any changes were made to the inputs, outputs or tunable parameters. Several bugs resulted from failing to update these structs correctly. These situations could have been avoided by better communication.

4.3 Communication

The basis of the communication architecture was the design from the previous year. The only design choice made by the team was the communication method between the two PCs, which was CAN. In addition to CAN, the other communication methods in use were RS-485, RS-232, USB ports and Ethernet. The robot had its own local network. The communication architecture can be seen in the figure below.

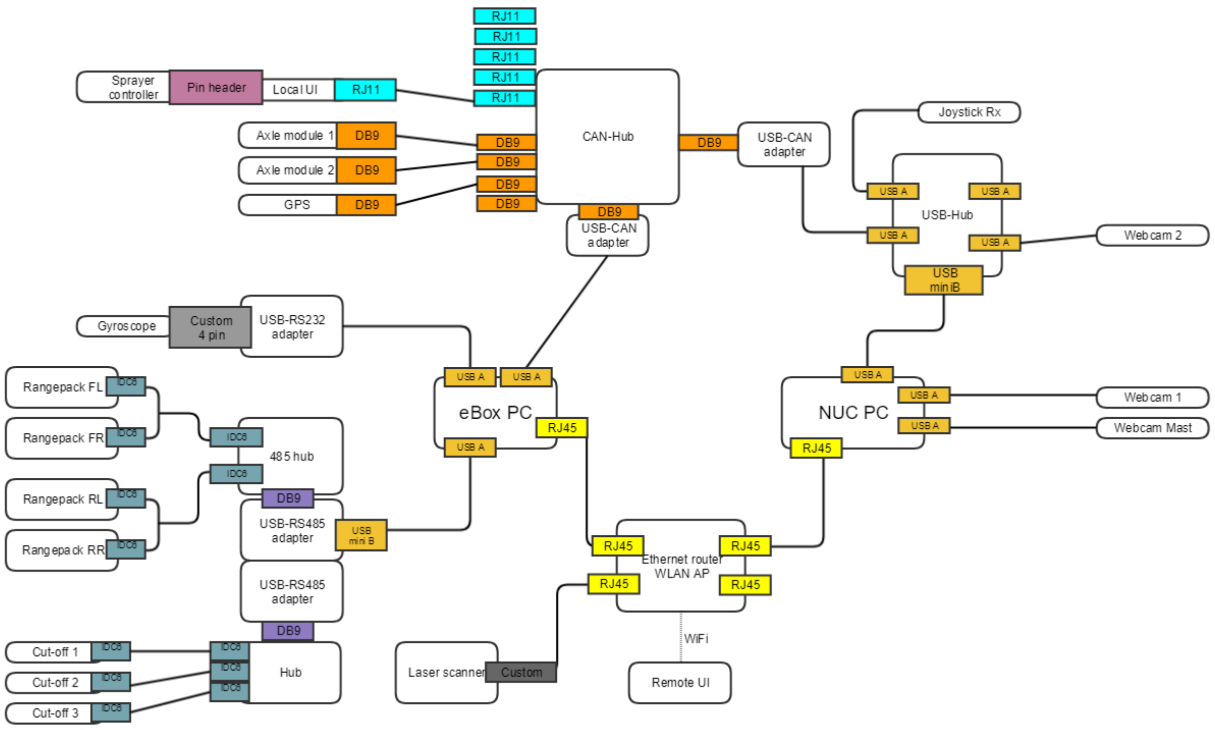


Figure 21: Communication architecture of the robot.

4.3.1 CAN

The purpose of CAN bus was to provide a common communication channel between the PCs and the peripherals.

CAN messages consist of the Arbitration, the Data, and the CRC fields as well as an Acknowledgement Slot. [8] The CAN 2.0b SAE J1939 was used with its arbitration field of the 29-bit identifier to assign priority to various messages and providing information about the sending device. The identifier for each message was constructed using following equation with gyroscope as an example:

Identifier = Priority + R + DP + PF + PS + SA = 29-bit (= 32bit)

Identifier component	*Range*	Gyroscope - example
Priority + R + DP	1(highest) - 7(lowest)	5 = 0x14
PF (PDU Format)	0xF3	0xF3 = PDU2
PS (PDU Specific)	0x1X - 0x8X	0x7X
SA (Source Address)	0x1X - 0x8X	eBox: 0x8A
PGN (PF + PS)	0xF3 + 0x1X - 0x8X	0xF371

Table 3: CAN Identifier Breakdown

The Data field included the relevant message data contained in eight bytes of data, such as the following Table presents for gyroscope:

Bits	Gyroscope - example	
0 - 7	Mode	disabled(0) or enabled('K')
8 - 23	Roll angle	0 - 65535
24 - 39	Pitch angle	0 - 65535
40 - 55	Yaw angle	0 - 65535
56 - 63	Data counter	0 - 255

Table 4: CAN Data Breakdown Example

These messages were formulated using similar standard as used in the previous year by determining the identifiers for each message to have relevant priority level and be exclusive from the other messages. The data field was built from the relevant information backed to convenient transportation format such as bit masks for boolean variables.

CAN messages were created for both reading from peripherals, such as the Local UI, as well as commanding them from the NUC PC with the latter type having higher priority over the former one.

These messages were configured to a database of the CanTrace software. This software provided live stream of packages along the CAN bus when connected to the on-board CAN board. The attached database then parsed the incoming messages for meaningful variables and names based on the configuration of those messages.

CanTrace software proved beneficial also determining if debugging issues were related to CAN communication or not, providing some trends of the data over time measurement in real-time with graphs, and manually commanding axle modules when testing their response.

The communication between the PCs could have been established with an IP based communication method, such as UDP. When the communication between the computers was discussed, CAN communications had already been implemented and were working. Since the message traffic from eBox to NUC did not cause the capacity of the CAN bus to be exceeded, CAN was chosen as the method of communication.

4.3.2 UDP

UDP communication was used between the machine vision program, the main program running on NUC and the remote user interface. The messages between the machine vision program and remote user interface were routed via the main program on NUC. The machine vision program received the timer step counter from the program on NUC and sent the results from the machine vision algorithms, which were used by the program developed in Simulink. The remote user interface received the input, output and tunable parameter structs as well as some other information from the program on NUC. The structs were serialized into bytes in order to be sent via UDP.

The downside of using UDP is that the protocol is connectionless and does not verify that messages are sent and received successfully. Therefore, it is possible that some messages are lost. Nevertheless, UDP was chosen because it was simple and easy to use. Losing an occasional message was determined not to be a significant risk for the functioning of the robot.

4.4 Simulink Architecture

There are three main blocks in the Simulink architecture: CAN to SI converter, logic and SI to CAN converter. Since all the logic is done using SI units the converters simply convert the input sensor and output actuator CAN messages to proper data type. Naturally, the input and output layers contain all the required signals for the CAN communication. Also the calibration of all inputs is done in the CAN to SI converter block.

The actual logic for autonomous control is implemented in the logic block after the sensory inputs and user-given controls have been converted into SI units. The autonomous control block contains a central stateflow chart, that is used to monitor and control the robot’s state. Also the numerous algorithms for navigation, positioning and other tasks lie inside the autonomous control block. All algorithms can be used interchangeably as they share the interface, making the architecture very modular.

In the central stateflow there are five fundamental states between which the robot operates: INIT, STOP, INROW, TURNING and TASK3. The robot begins from the INIT state and drives forward until it senses a maize row. Then, it begins INROW state where it will navigate until it either sees an obstacle during Task 3 or Task 4 or end of maize row. TASK3 simply activates upon an obstacle is detected and the robot goes back into INROW state after the obstacle is passed. End of the maize row detection enables the turning system that executes the current element of the turn pattern. After that the turning algorithm ensures that the robot is again inrow and the central stateflow turns the state back to INROW.

There is also a possibility to stop the robot at any time using the remote user interface (RUI), joystick or LUI stop button. Then, the last state index is saved and the robot state changes to STOP. The state will then go back where it was after START button has been pressed from any of the user input sources mentioned.

The architecture is built so that it supports changing the robot state while the robot is in STOP state. This was due to the competition rules. This mechanism is later illustrated in the turn algorithm chapter.

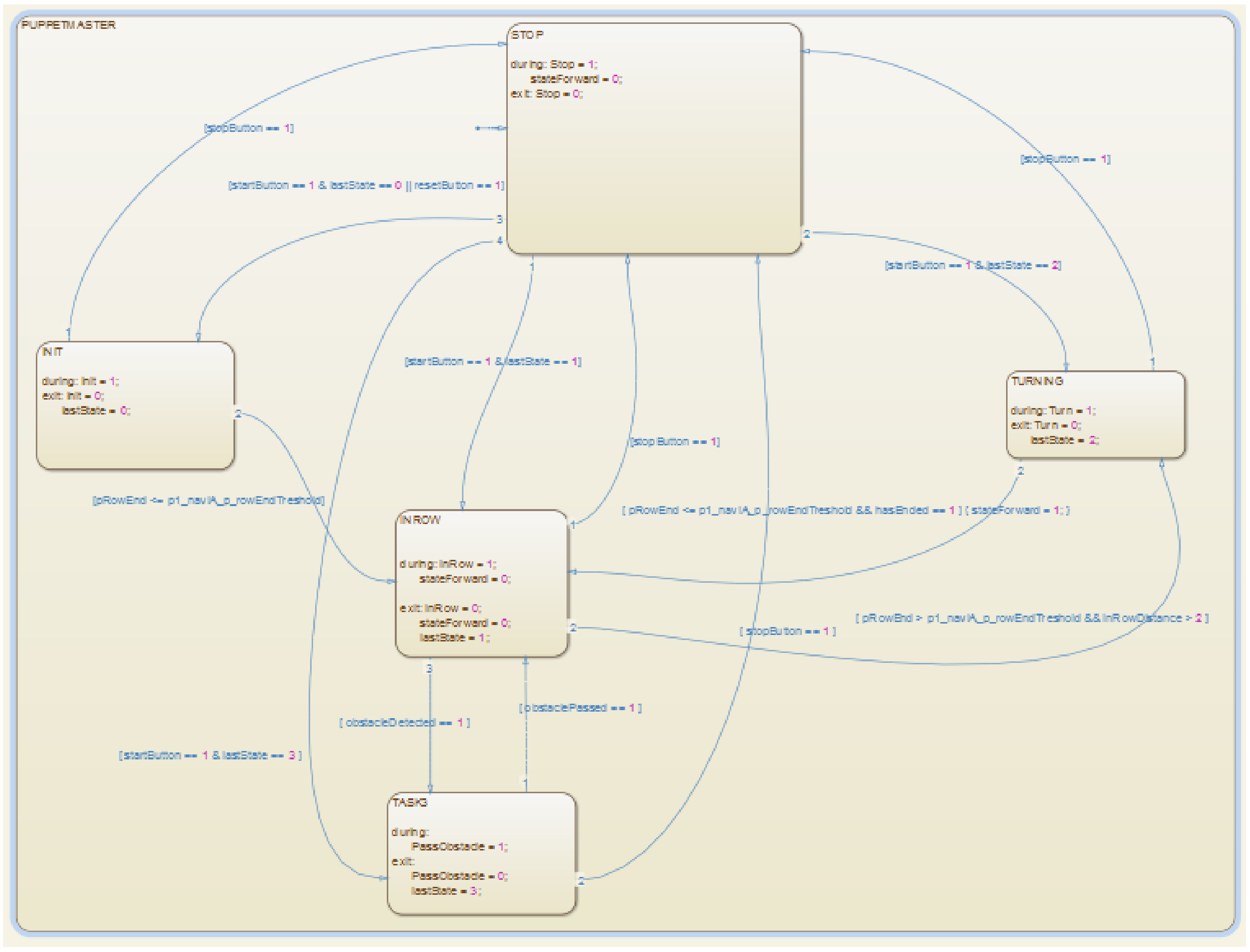


Figure 22: The central state machine of the robot logic.

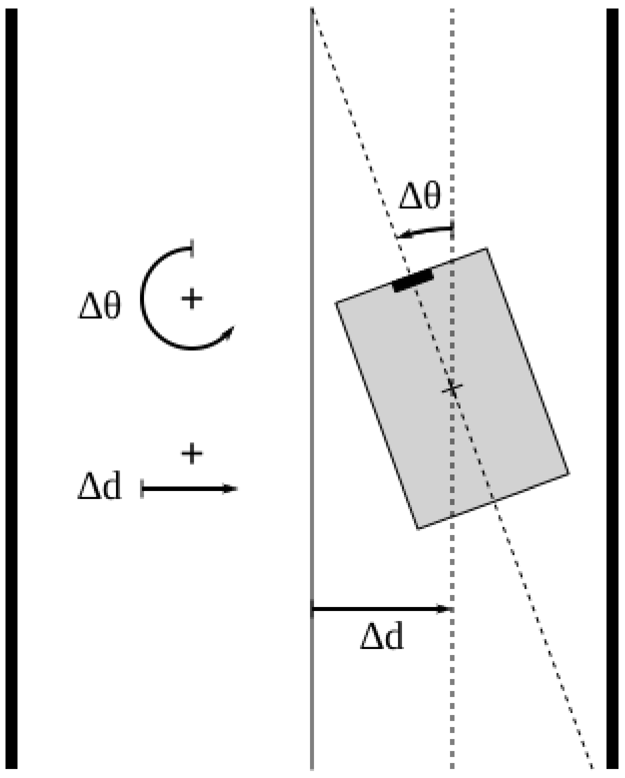
It is possible to to drive the robot manually as well. Enabled subsystems were used to choose whether the robot's control signals are chosen from the manual or autonomous logic block. Additionally, enabled subsystems save resources as the disabled blocks are not executed.

The mechanism is also used in the various alternative algorithms used in positioning, navigation and choosing tasks to be executed. The central state machine outputs the enabling signals into the numerous enabled algorithm blocks depending on the robot state.

4.5 Positioning

The main goal of positioning system is to provide the navigation algorithms information on the robot's orientation and location relative to the surroundings. Moreover, the navigation algorithms require the robot's angular and lateral error measured from the center of the maize row. The definitions for the error directions in the coordinate frame can be seen in Figure 23. Another critical task for the positioning is row end detection. It ensures that in the end of the row the central state machine gets a reliable signal to set the robot to TURNING state.

Different positioning algorithms used RangePacks or laser scanner (and machine vision). In the next paragraphs the arsenal of positioning algorithms are explained. In order to survive from gaps up to one meter in the maize row the robot needs some kind of history of last known states. This is discussed in the following paragraph.



Where

$\Delta\theta$ is the angular difference from the center line,

Δd is the lateral difference from the center line

Figure 23: The coordinate system for positioning errors.

History C

The history C positioning algorithm uses plant distance data from ultrasonic RangePacks to create history data to build history of the robot distance from the maize field to drive regression analysis, that can further refine the data into robots pose in the field.

Four different RangePacks provide data for left and right, front and left RangePacks. The raw ultrasonic data is converted to SI units and filtered from long distance findings. Robot keeps track of one second accumulated data from RangePacks to calculate the position. With control algorithm running 25 Hz, this means 25 previous records of each RangePack is kept in the history data. The history data consisted of XY points of the last measurements data. On each iteration, the data points were rotated and moved to correspond to the robot movement. This is realized by simple affine transformation matrix, that both counted for the heading change as well as x and y-wise movements.

In Figure 24, the RangePack data of robot driving is drawn first as seen by robot, and below this, the angle and distance by regression information is plotted (as extracted from the history data points). To compensate more over missing plants in the field, the data points in history data were moved closer to middle of the robot centerline, otherwise missing plant would drive the regression more over compensate towards the other direction. In Figure 25, the history data seen by robot is visualized in simulated test field. Due to the nature of the RangePack is collected, discontinuation of the field plant row symmetry seen in this figure is hard to compensate when using ultrasonic RangePacks for localization.

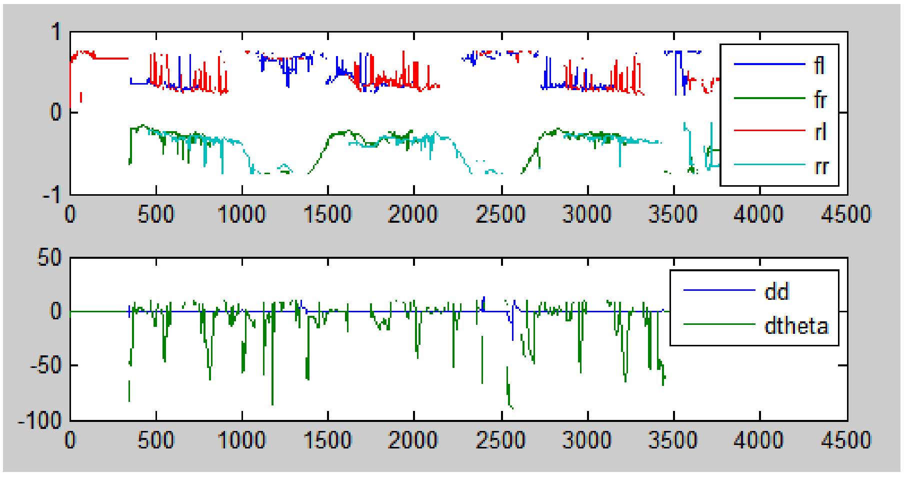


Figure 24: Sample of RangePack data and navigation data visualized together with the derived positioning data during the tuning process.

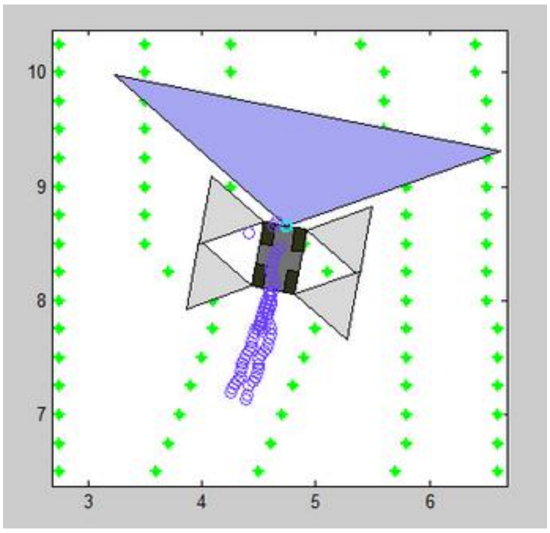


Figure 25: History plotted behind the robot, this works well unless the plant row pattern is disrupted as in test field called 'aplesiini' seen in the picture. Test fields were used to test and tune the behaviour on different scenarios, even tho, some of the fields tested were later considered unrealistic for the competition.

Positioning A: Using RangePacks

The positioning algorithm A is based on the RangePack history data from History C. A linear least squares regression fit was used to estimate the center of the row from the RangePack history data. Once the history points were translated to the center of the row, the regression algorithm was used to obtain the best linear fit in the least squares sense. This center estimate was then used to approximate the angular and lateral error of the robot center and heading to the row center line.

The regression algorithm was written as an individual embedded Matlab function. Matrix algebra formulation was used. In addition to the basic linear least squares solution, some possible problematic situations were considered. For example to avoid any inversion problems with the Gramian matrix $X^T X$, a very small constant was added along the matrix diagonal to ensure the matrix is positive-definite.

In addition, a few different options were studied in order to calculate the goodness of fit, including the r-squared value and normalized RMSE. It was originally intended that the goodness estimate would later be applied in speed control, however, none of the implementations was enabled in the eventual solution. This was mostly due to problems with the r-squared value when the linear fit is parallel to an axis, i.e. the variance of the dependent variable is not related to the independent variable.

Positioning C: Using the laser scanner

The positioning algorithm C is based on distance array from laser scanner. The distances were transformed into robot's xy-coordinates (figure 26 gray dots) in the SI-converter block. The main idea of the algorithm was to rotate and move sideways the coordinate matrix in order to find the optimal match based on how many coordinate points were in certain areas.

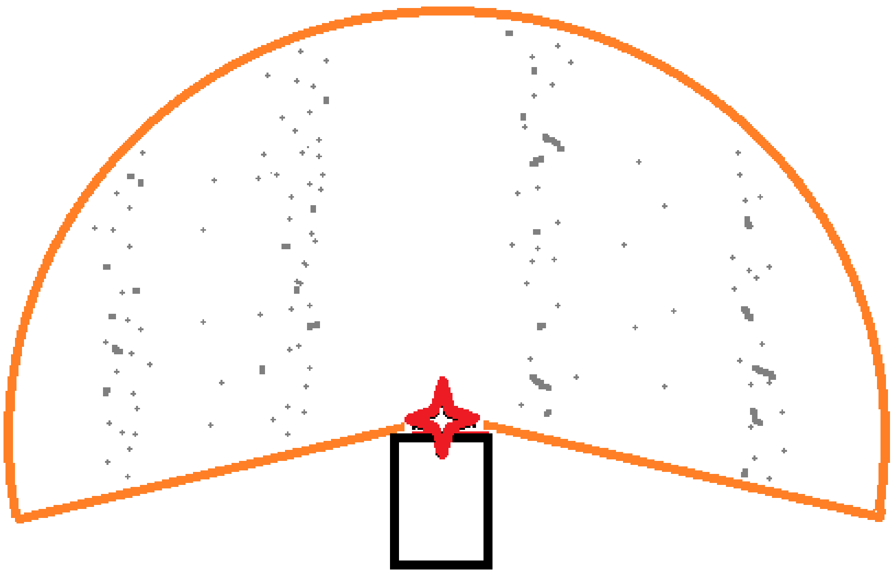


Figure 26: Laser scanner field from inside of a maize field. The orange line reflects the range of the laser scanner that we had defined (diameter 2m). The gray dots represent laser scanner data and the form of 4 maize rows can be seen in it.

The rotation was performed with rotation matrix and in the first iteration rotations from -50 to 50 degrees with resolution of 10 degrees were made and sideways iteration was performed from -0.2m to 0.2m with resolution of 0.04m. The second iterations were made around the first best values from -5 to 5 degrees with resolution of 1 degree and from -0.02 meters to 0.02 meters with resolution of 0.01 meter. When calculations were made in two loops considerable efficiency improvement was achieved.

The optimal matches were determined by first determining four boxes, that can be seen in figure 27, so that the middle point of the next box was one row width away from the previous. The goodness of the fit was simply determined by the total amount of data points inside those four boxes. When the goodness for all rotations were calculated, the rotation and lateral errors were the same as rotation and sideways shift of the best rotation (figure 28).

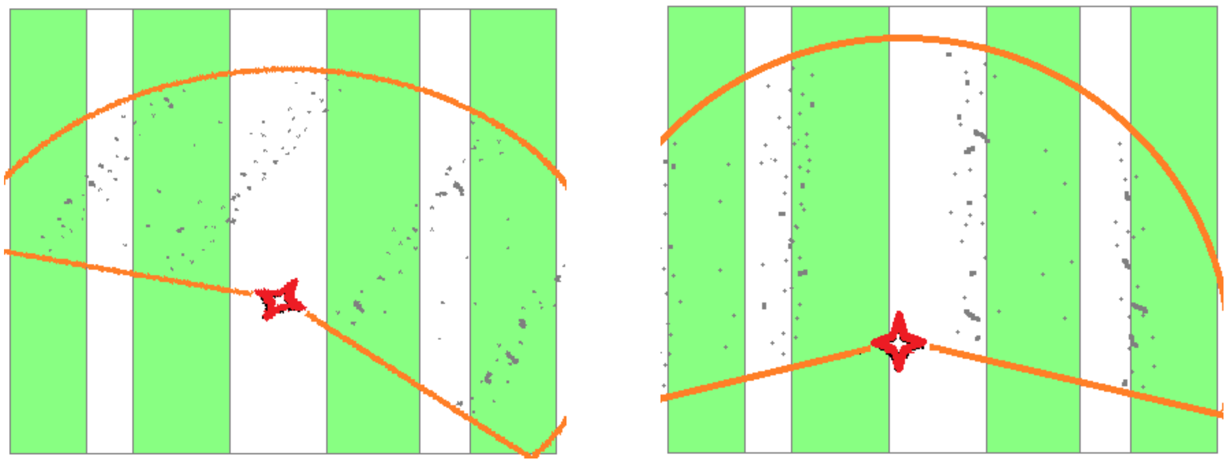


Figure 27: Green color represents the four acceptance boxes in which the amount of data points (gray) were maximized by rotating (left image) and sifting the points sideways (right image)

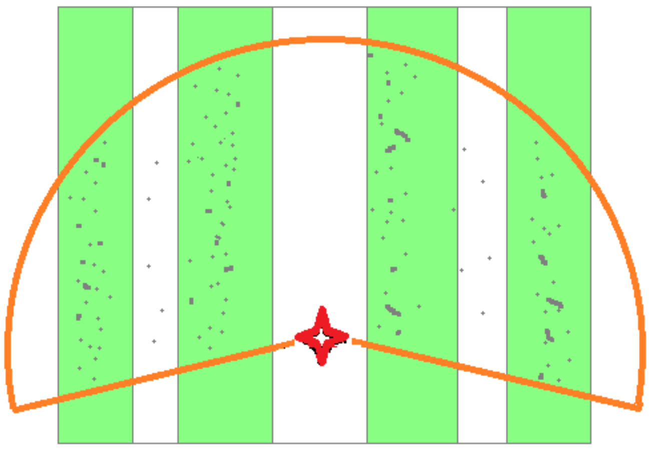


Figure 28: An example of the optimal match found. The rotational and lateral errors were equal to the rotation and sideways shift of the optimal match compared to the original data. In this figure both errors would be close to zero.

This was the most efficient and reliable of the implemented positioning algorithms, hence it was also used in the competition. Even tests with velocity slightly over 1 m/s inrow speed were successful in the test field. One of the main benefits compared to RangePack based ones is that after the turn it can start inrow navigation mode even before the robot is actually in the row.

Row end detection A

The row end detection A algorithm does not engage the history but uses the current RangePack readings to indicate a probability for the row end at each time instant. If the probability rises over a given threshold it is deduces that the row end has been reached.

Pseudocode:

```
Initialize Probability to 0

If no RangePacks sense maize
    Increment Probability
Else If both front or both rear RangePacks do not sense maize
    Increment Probability
Else If all RangePacks sense maize
    Decrement Probability
End

If Probability > threshold
    The row has ended
Else
    The row has not ended
```

Even though this basic algorithm performed well it was not used in the competition. This algorithm was highly dependent on the robot speed. Thus, it should have been fine tuned for the competition ground and speed before the first task. The downside of this algorithm is that since it has to ignore gaps up to 1 meter in the maize row it will also drive that distance into the headland before noting the row end.

Also it is trivial why the laser scanner performs better than this basic algorithm in the basic forward drive. The laser scanner is pointed forwards where the RangePacks are pointed perpendicular to the robot's heading. Thus, laser scanner can *predict* and RangePacks can only *observe* the current state. Interestingly, there are two sides to this seemingly trivial choice of algorithm. Even though the laser scanner proved superior over RangePacks when driving forwards, they could still have been used for reverse drive as the only one laser scanner was placed in the front of the robot.

Row end detection B

The row end detection B algorithm utilizes the history and calculates the amount of history points inside a box. The box width is set to row width for convenience and box height is a tunable parameter. The amount of points inside the box is then divided by a value that is gathered by calibration of the algorithm. The ratio is then the probability for row end.

This algorithm has the same advantages and disadvantages as any RangePack algorithm. This one is slightly easier to tune than the Row end detection A -algorithm as this has more straightforward relation for the dependency of parameters and output. However, this algorithm was not used during the important last tests nor the competition. The performance of this algorithm was satisfactory, however, not the best available option.

Row end detection C

Row end detection C was based on laser scanner data. The row end probability was calculated by dividing the amount of misses in the specified scanning area aka infinite values in the data, by the maximum amount of data points in laser scanner data.

This algorithm's only weakness compared to the RangePack based ones was that it would not be utilized in reversing because when moving to that direction it could only notice the row end when the row had actually ended 2 meters ago.

Advantages of this algorithm is that it noticed the row end immediately when the laser scanner reached the end of the row and it had no problems with inrow gaps since the laser scanner data can see over 1,5 meters forward. Additionally, after a turn the start of the row could be detected even before the robot was actually in the row. This algorithm was used in the competition setup.

4.6 Navigation

The main goal of navigation is to provide the right control signals to the axle modules during INROW and TURN states. Several algorithms were developed for both states.

4.6.1 Inrow

During *inrow*, navigation gave such control signals to the axle modules that minimize the positioning error. A PID controller with feedforward was used for this. The angular error was fed straight to wheel angles to counter the effect of drifting. The feedforward reacted faster than the bare PID and it did not have the effects of integrators.

In order to avoid any errors caused by the integrator windup during turns for example, a basic anti-windup was implemented by disabling the integration and feeding zero error during turns.

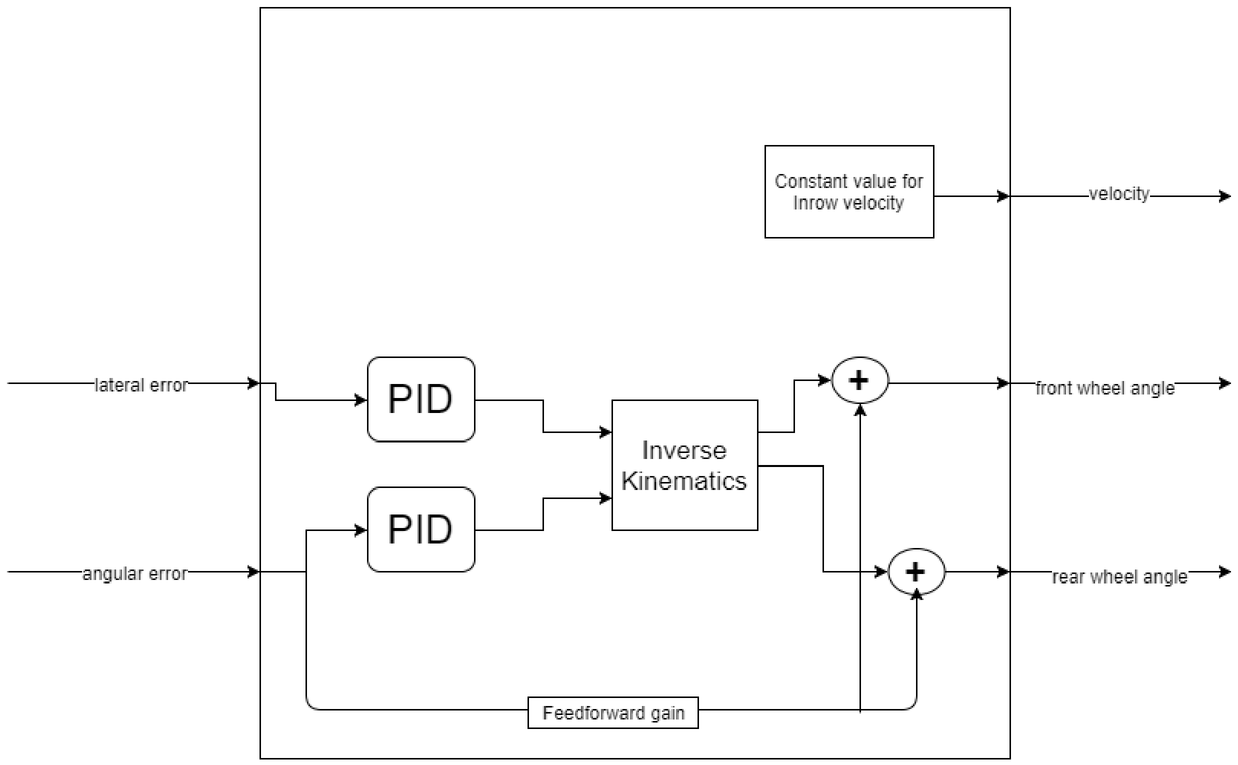


Figure 29: The navigation block.

The PID controller output was then fed to inverse kinematics to provide the desired control signals for the wheel angles.

4.6.2 Inverse kinematics

The analytic solution of the inverse kinematics of the system is challenging, and thus an approximative approach was used. Here, the approximate inverse kinematics is a mapping from lateral and angular speed ($\dot{x}$ and $\dot{\theta}$) to the front and rear axle angles (φ_r and φ_f) based on polynomial functions. The functions were estimated with linear regression.

The forward kinematics were used to make a mesh of corresponding values from front and rear axle angles to the lateral and angular speed of the robot center. Since the velocity is linear, it was set as constant when generating the mesh. This data, a mapping from a mesh of forward kinematics input values to corresponding output values, was used for the regression fit. The inverse kinematics were estimated with polynomial based linear regression up to third degree basis functions including constant and cross terms. The following parameter matrix X was used to fit both front and rear axle angle.

$$X = [\,1 \quad \dot{x} \quad \dot{\theta} \quad \dot{x}^2 \quad \dot{\theta}^2 \quad \dot{x}^3 \quad \dot{\theta}^3 \quad \dot{x}\,\dot{\theta} \quad \dot{x}^2\,\dot{\theta} \quad \dot{x}\,\dot{\theta}^2\,]$$

The solution to the linear regression is given by

$$\hat{\beta} = (\,X^T\,X\,)^{-1}\,X^T\,\varphi$$

where φ is either φ_r or φ_f .

This results in different coefficient vectors β_1 and β_2 for rear axle module angle and front axle module angle

$$\varphi_r = \beta_{10} + \beta_{11}x_1 + \beta_{12}x_2 + \beta_{13}x_3 + \beta_{14}x_4 + \beta_{15}x_5 + \beta_{16}x_6 + \beta_{17}x_7 + \beta_{18}x_8 + \beta_{19}x_9$$
$$\varphi_f = \beta_{20} + \beta_{21}x_1 + \beta_{22}x_2 + \beta_{23}x_3 + \beta_{24}x_4 + \beta_{25}x_5 + \beta_{26}x_6 + \beta_{27}x_7 + \beta_{28}x_8 + \beta_{29}x_9$$

These polynomials were used as a part of the navigation algorithms to approximate the inverse kinematics of the robot.

4.6.3 Turning

In the development phase several different prototypes for the robot turn were done. As the four wheel steering minimizes the turn radius to as little as 0.75 meters there was no need to make any complicated turn maneuvers in the beginning.

The turn may contain skipping rows in Task 2. First, odometry was used to skip the right amount of rows. As it failed often, a turn algorithm that consists of many states that help the laser scanner to calculate skipped rows robustly was implemented. Even though even thousands of precious milliseconds may have been lost with the failsafe algorithm that consist of several substates, the performance was flawless in the end. The base of the turn is as follows:

1. Detection of the end of a row triggers the turn
2. The robot rewinds a bit with row ends A and B / the robot drives forward with C
3. The robot turns 90 degrees to the desired direction
4. The robot rewinds a bit
5. The robot skips rows if necessary while navigating on headland
6. The robot turns 90 degrees to the desired direction
7. The robot drives forward until it senses the next row

Because the row end was detected significantly later with RangePacks compared to the laser scanner, the robot had to compensate this error by driving forward or reversing based on wich row end algorithm was used. Then the robot turns 90 degrees to desired direction and reverses a bit to make counting skipped rows easier and more robust. At this point the headland drive mode is activated and the robot starts to calculate row ends for row skipping purposes while simultaneously keeping constant distance to the field. When the desired amount of rows is skipped the robot turns another 90 degrees in the same direction as the first time. Lastly, the robot drives forward until it senses the row. However, with laser scanner based navigation it senses the row immediately after the turn so this phase is not required.

The gyroscope was utilized in the turn to measure the turn angles. With slow enough speeds it performed well enough. It was planned to make the turning faster in the beginning and slow it down when the angle is near the target as the gyroscope output changes slowly. This was never implemented.

Stateflow was used to interpret and program the turn as a state machine. A transition could then be made of each state to the next and previous states to allow changing the robot's turn pattern state while the robot is stopped as per in the competition rules. If *state rewind* -button is pressed and the robot main state is STOP, the turn state is rewinded. State forward is done similarly.

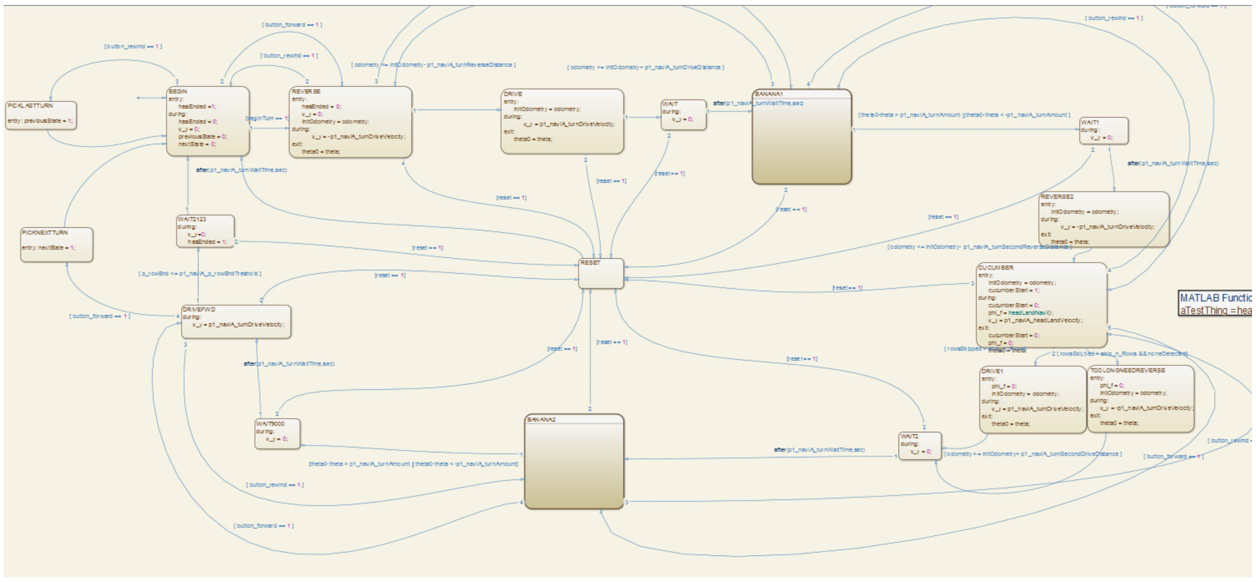


Figure 30: The turn state machine.

4.6.4 Headland drive and row skipper

The headland drive had a very simple repulsion logic to keep the desired distance to the field; If the laser scanner detected something in front of it, it turned a bit away from the field and if there were not enough data points on the field side of the robot it turned a bit to that direction. This solution worked robustly but some more developed one could have been more reliable if there would have been larger differences or damages on maizes at the end of the rows.

The row skipper used basically the same algorithm as positioning C, but this time it had a fixed 90 degree rotation and only lateral error were calculated from 0 to row width. This resulted in a quite clear sawtooth signal (figure 31) that was used with odometry to calculate how many end of rows had been passed. The odometry was used to prevent miscalculations since sometimes the signal could bounce couple of times when row end was detected.

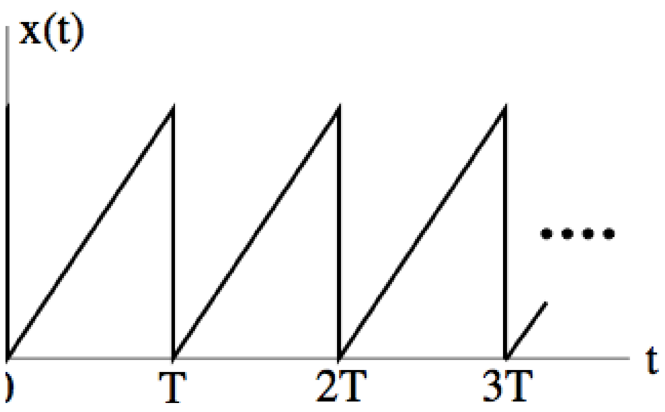


Figure 31: An example of the row skipper's sawtooth signal. The x(t) would rise from 0 to the row width and when laser scanner reaches the beginning of the next row the value would drop to zero and start rising again when the robot moves forward.

4.7 Task 3 Software Logic

In task 3, pink and yellow golf balls represented weeds and obstacles on the field. These items needed to be detected and each detection signaled.

Since Task 2 and main architecture logic took more time than expected, simplifications were made to the other tasks. Headland obstacles were ignored as the case is very difficult and includes many alternative responses depending on the surroundings. If an obstacle was detected, the robot would not come back to that row even though there could be weeds behind the object. The robot would enter each row once and continue to the next one to visit as many rows as possible and detect as many objects as possible.

The plan was to use a three camera setup to detect objects from three rows parallel. Since the competition rules required to indicate which row the detection was made in, the mast leds would have been used to indicate the row in question. In addition, the siren would have been used to indicate all detections.

Task 3 was realized as an interrupt for task 2. Upon an obstacle detection the central state machine switches into Task 3 state. This caused the robot to do the following

1. Stop and toot
2. Rewind until row end detection
3. Perform a crab turn into the next row
4. Continue exploring in Task 2 INROW-mode

The horn logic ran a small stateflow parallel with the task 3 block. It simply output a boolean true to the horn for 2 seconds upon a weed detection and 5 seconds for obstacles.

All individual parts but object tracking in machine vision was completed separately. However, the machine vision, tracking or Simulink logic were never integrated successfully.

4.8 Task 4 Software Logic

In task 4, the weeds on the field needed to be sprayed. This task implementation also included simplifications. First, it was attempted to make a representation, such as a map, of the detected weeds to spray but as the project time ran out, it was decided to do task 4 as an extension to task 3. Task 4 is the same as task 3 with the sprayer logic enabled. As far as the team understood, this would have fit the competition rules for task 4.

Upon a weed detection, the odometry calculation would be initialized and the sprayers were immediately turned into the right direction. When the current odometry value was then at the point where the sprayer was at the weed, sprayers were turned on for a short pulse-like instance. This logic was done with a stateflow chart.

Since task 3 was not finished it affected task 4 workflow as well. The sprayer hardware and drivers were completed and the Simulink logic was finished during the competition. However, with no working task 3 the functionality was never tested.

4.9 Simulator

Before any logic was tested online or before there were even proper conversions, calibration, hardware or communications the robot logic was unit and integration tested in a simulator environment. An s-function was set up in simulink that drew the robot, its sensor ranges and a scenario with maizes. The s-function got the robot pose from the kinematics block. The maize scenario was parsed from a txt file. RangePack data was simulated by calculating the distance from robot corners to the maize. laser scanner data was simulated by first determining the mazes that are within 4x2m box and then returning their xy-coordinates similar to laser scanner data after SI-conversion.

Some delay and noise models were added to increase realism. The noise models were utilized especially in maize field generation and odometry. However, more noise models could have been implemented to make simulation more accurate, but even as it was it served its purpose sufficiently. The delay of 40 milliseconds corresponds to the robot's duty cycle. Uniformly distributed noise was added to the simulated RangePacks with some parameters.

The simulator could then be used to roughly tune the navigation PIDs and positioning algorithms. Most importantly valuable information was gained on the algorithm bugs and performance issues without driving an inch online.

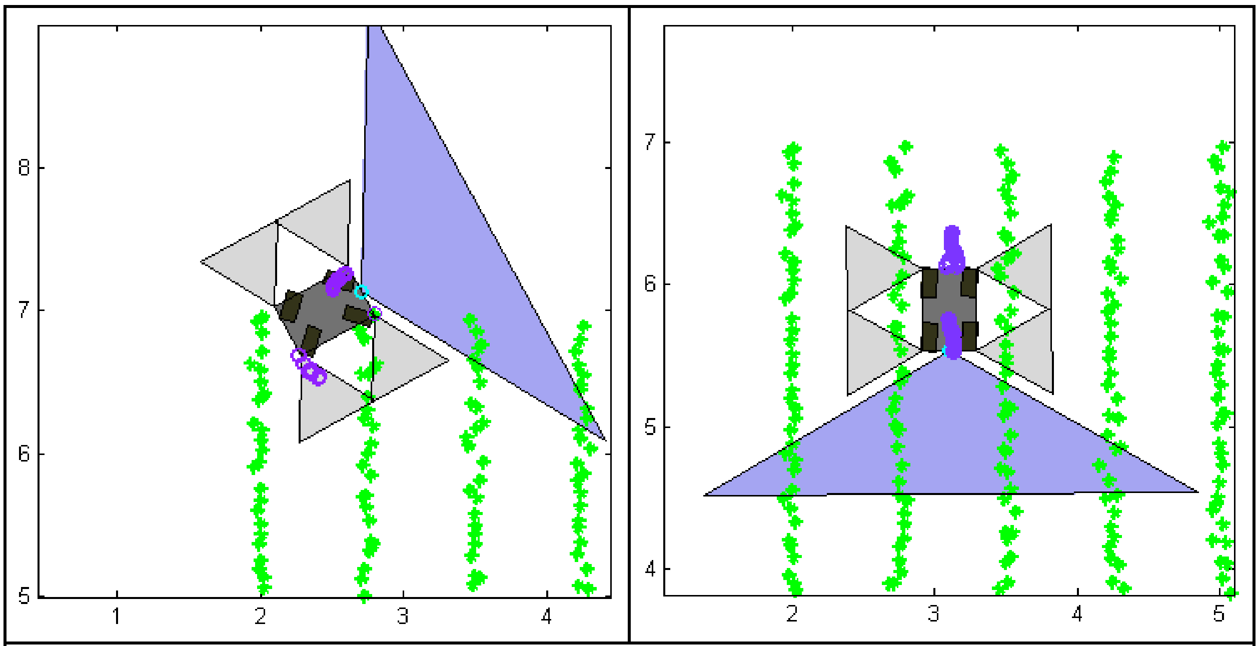
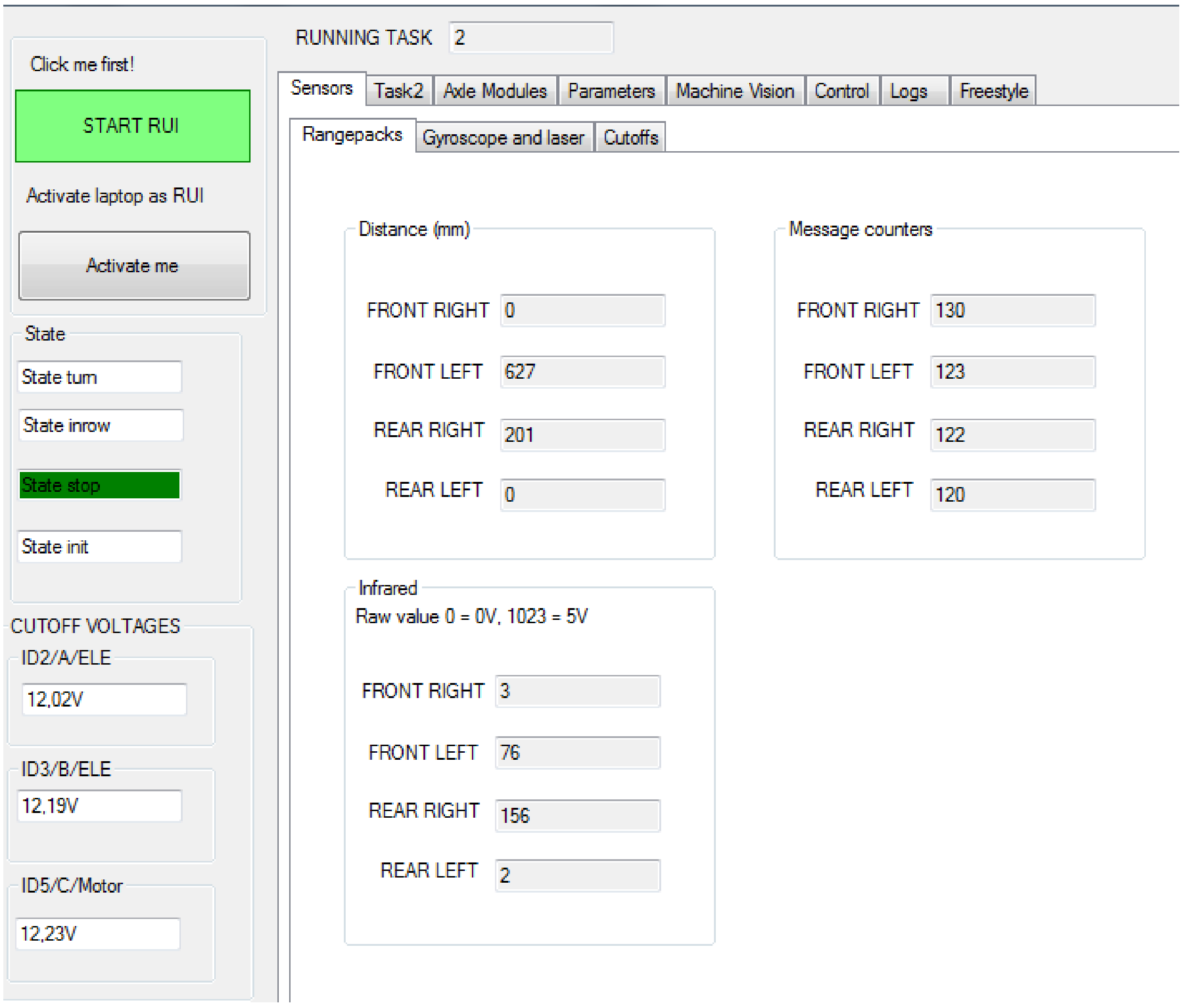


Figure 32: Example images of the robot simulator in TURN and INROW modes.

4.10 Remote User Interface

The remote user interface (RUI) is a GUI program that could be run from any computer connected to the robot's network. The program was implemented using Microsoft Visual Studio's Forms. The program communicated with the main program on NUC using UDP. The programs shared a dynamic-link library containing the UDP port definitions, classes for serializing the messages and struct definitions for the Simulink inputs, outputs and tunable parameters. The purpose of the remote user interface was to allow monitoring the state of the robot as well as provide additional functionalities. These functionalities included starting and stopping the robot, parameter tuning, saving and downloading parameters as XML files and editing the log file name and setting the turn pattern for the advanced navigation task. The Simulink inputs and outputs received from NUC included all the data from the sensors, such as the voltage readings from the cutoffs.



RUNNING TASK 2

State

State turn
State inrow
State stop
State init

CUTOFF VOLTAGES

ID2/A/ELE	12,02V
ID3/B/ELE	12,19V
ID5/C/Motor	12,23V

Distance (mm)

FRONT RIGHT	0
FRONT LEFT	627
REAR RIGHT	201
REAR LEFT	0

Message counters

FRONT RIGHT	130
FRONT LEFT	123
REAR RIGHT	122
REAR LEFT	120

Infrared
Raw value 0 = 0V, 1023 = 5V

FRONT RIGHT	3
FRONT LEFT	76
REAR RIGHT	156
REAR LEFT	2

Figure 33: The remote user interface.

4.11 Collecting logs

Log files were collected of the whole system data including parameters, Simulink inputs and outputs, such as sensor readings, and all other necessary data. The log files were used to reproduce the behaviour of the robot system in a computer environment. The data was collected in separate files for the parameters, the Simulink data and the laser scanner data. These files were imported to Matlab and input into a Simulink model to simulate a drive with the actual robot. The simulation was mainly used to simulate manually operated test drives, and the simulation result was used to validate and tune the algorithms and parameters, as well as to uncover and analyze any unexpected or faulty operation. An example of some of the simulation result plots is presented in Figure 34. Another simulation model was made to replay an autonomous drive from the log files.

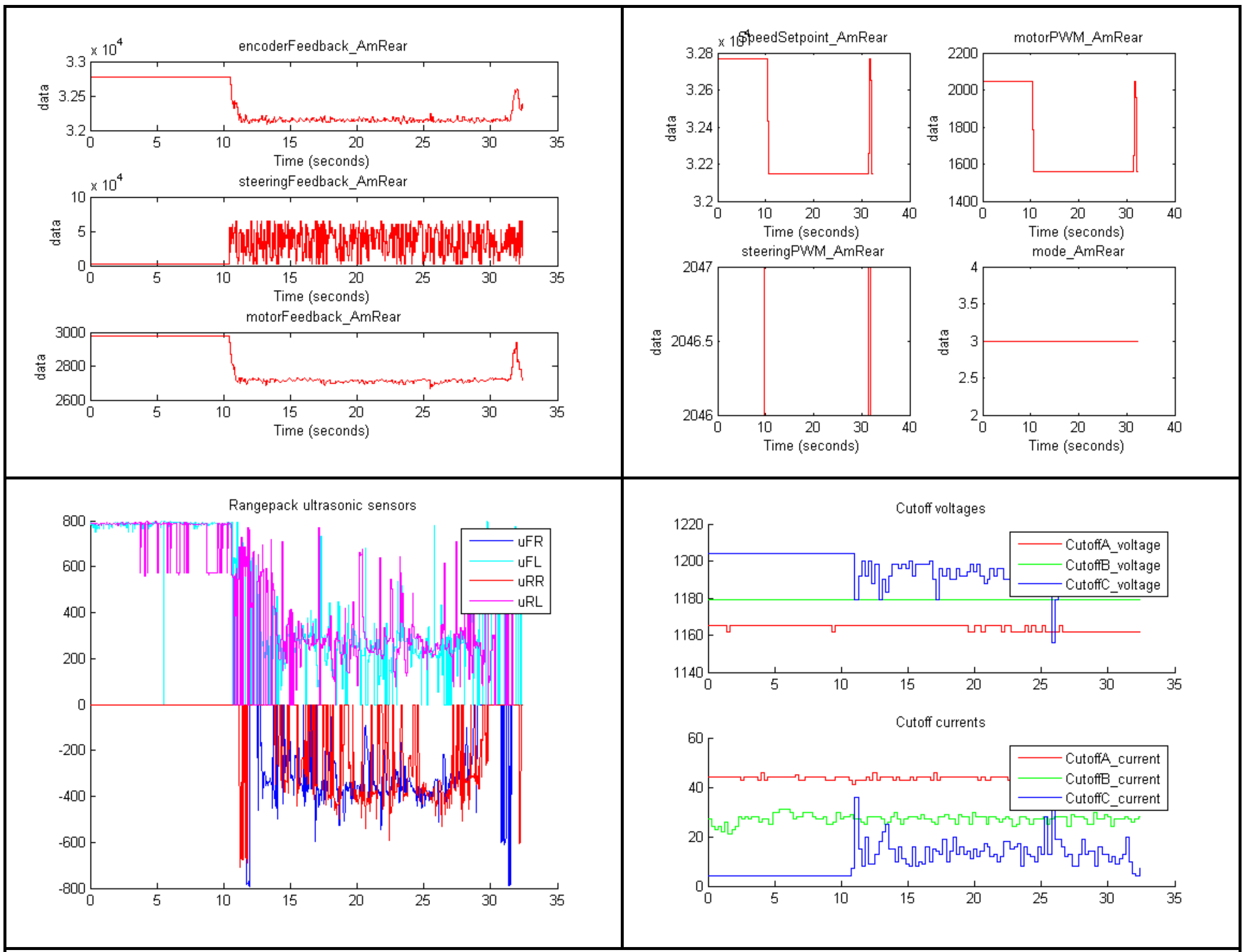


Figure 34: An example of log simulation result plots from a manual test drive on 13.6.2017 at the Harper Adams soil hall test field.

Various manual test drives were specified beforehand to serve the purpose of algorithm tuning. Different test drives were specified for tuning different algorithms and parameters. In the test drive specifications for example the following parameters were controlled: robot constant speed, robot lateral and angular offset from the plant row center, and the drive distance from the row end. In addition, machine vision algorithm tuning required camera image logs, for which the placings of yellow tennis balls (obstacles) and pink golf balls (weeds) were defined. For documentation, also environmental factors were recorded, such as the plant row length and distance between rows, as well as plant height.

The test drives were driven manually on a test field with a joystick controller according to the specifications. Each drive was documented with a predefined form. All log files were gathered at the end of a test drive session and archived with the corresponding documentation. These log archives were used for systematic algorithm tuning, especially in new environments. One of the most important use cases of the log files was in particular the quick tuning of the algorithms in previously unknown environments with different characteristics. It is a powerful tool since it allows the initial tuning to be performed without driving the robot autonomously in the test field. However, some accustoming to the workflow was required to fully utilize the potential of the log based algorithm tuning process.

5 Machine Vision

Machine vision was implemented for tasks 3 and 4, as well as for an additional row detection and positioning algorithm.

5.1 Camera Parameter Tuning

The web cameras have various parameters that affected the performance of machine vision algorithms. Therefore, these camera parameters are tuned for the given environment conditions before any tuning of the machine vision algorithms.

The camera parameter tuning involves physical parameters, such as the camera angle, and camera settings, including the white balance and the exposure time. The physical parameters are tuned to a desired viewpoint by fixing the camera position, as well as the vertical and the horizontal angles. The camera settings are tuned with combination of OpenCV functions and Microsoft Lifecam software, allowing live feedback from the camera view while using the setting sliders.

The basis for the setting tuning comes from seeing the clearest difference between background and the desired objects to detect as well as the robustness of the object tracking from robot motion. Once the machine vision algorithms are decided for the run, the clearest difference between objects and the background in the color transformed scene is seen as superior from the regular view. Therefore, the settings that maximize this difference are prefered. The tracking robustness is achieved by lessening the motion blur with smaller exposure time, while avoiding most of the shutter effect caused by insufficient exposure time.

Additionally, the tracked objects need to be seen in several scenes. Hence, the frames-per-second (fps) for image capture has to be sufficient while still maintaining efficient processing performance of the vision algorithms. This condition of 10 fps was satisfied when the Remote UI was not in use.

5.2 Log Gathering

The purpose of log gathering is to provide datasets for tuning the machine vision algorithms. To suit this purpose, the log gathering setups were planned in advance to isolate different test cases such as obstacle, weed, and mixed detection scenarios from each other.

The collected datasets reflected different lighting conditions, shaking, and other disturbances of the environment. The variation in these conditions enabled more robust tuning of the algorithm parameters as well as better comparison of the algorithm options for detection.

An OpenCV script was created for writing video feed from cameras to image files in combination with the step counter shared with the other log files. This synchronization of step counter was done to provide the system status at the moment of the image capture, for example the robot’s velocity for the tuning of the object tracking.

The collected datasets were then processed with LogReplay program, written in C#, to test and tune the machine vision algorithms. This process will be explained in the following chapters.

5.3 Perspective Transform

Perspective transform is required to remove the perspective projection innate to cameras’ images. This transform can be achieved by calculating the transformation matrix between the camera (pixels) and world (meters) coordinates. To obtain this matrix, several measurements are required, consisting of the camera height, h_{world}, and the distance to the lower and upper part of the picture, d_{world} and D_{world} respectively, as illustrated in the Figure below. These measurements are then used to calculate the vertical camera angle using trigonometry on the world side view. This angle is then used to measure the horizontal camera angle using assumption that the pixels are squares by excluding the lens distortions. The horizontal camera angle allows the examination of the horizontal dimension of the World view, which allows the measuring the horizontal coordinates of the World view. These horizontal world coordinates then provide the tools to measure the respective camera coordinates in the Camera view. The resulting four corner points are then used to derive the perspective transformation matrix.

Perspective transform was implemented with openCV functions by using the calculated perspective transform matrix from the four corner points to warp the given image to match the world view. After this transform, the images had straight maize rows which enabled the line detection algorithm to find straight rows and the detection algorithms to provide real world coordinates along with their object detection.

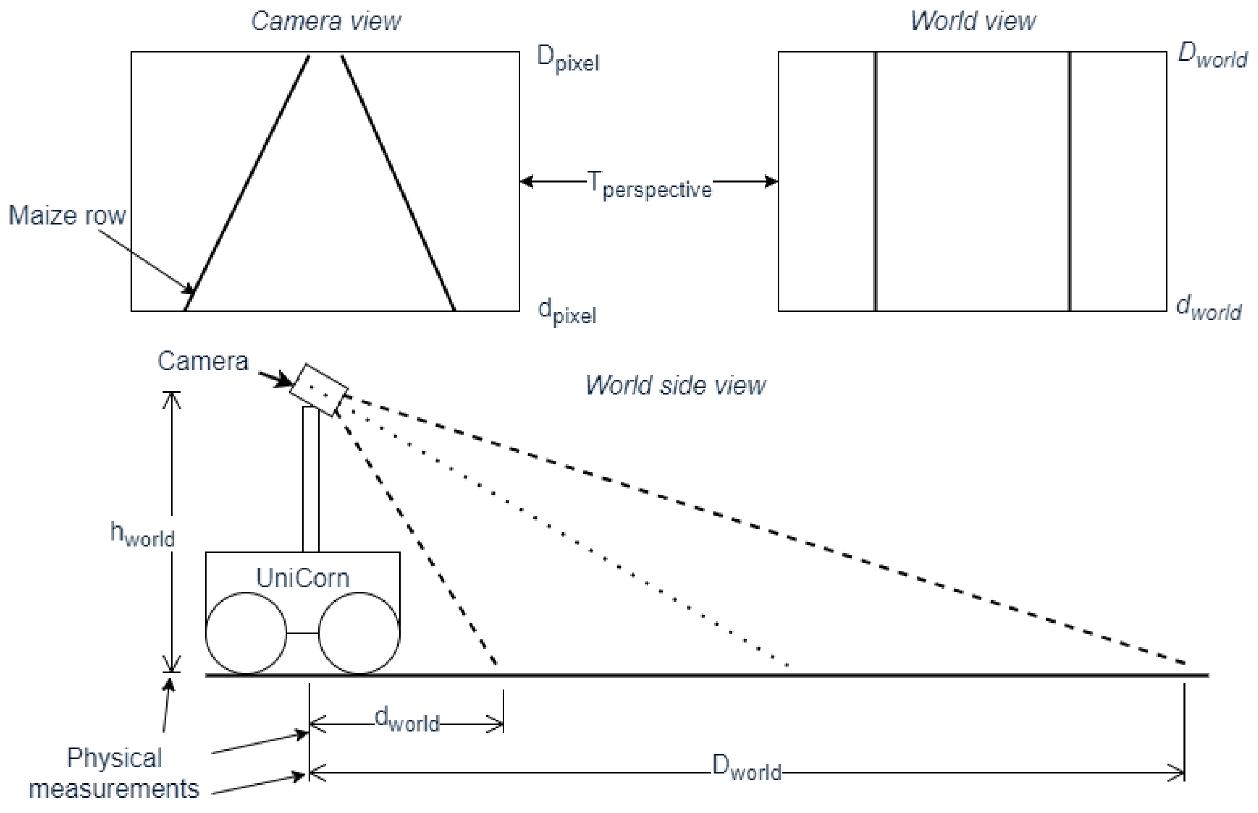


Figure 35: Three viewpoints used to derive perspective transform to the camera images

5.4 Color Spaces and Color Information Separation

As most of the interesting information to be extracted in the machine vision images is stored in the color of the pixels, it was feasible to approach the problem by finding suitable color spaces for detecting the positive matches. Several color spaces were discussed for the tasks, namely RGB, BGR, HSV, EGRBI and ECCI [9]. Each of the color spaces present the color information with vector space, but some give advantages for the data processing.

In most of the tasks, the pipeline started with separating the color information by thresholding color data. But depending on the vector space presentation of the color information, it made it more easy to detect the target colors.

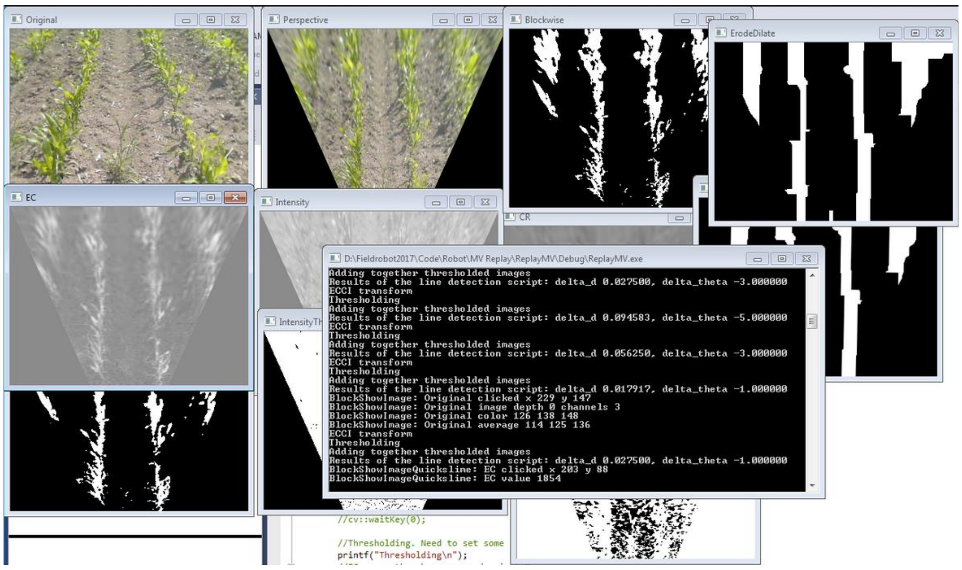
Excessive green red blue intensity (EGRBI) color space defines the color space base vectors as [2G R B] [(R+G+B)/3] and EC channel is a cross product of these, to keep the base vectors orthogonal. HSV transformation is different by mathematical nature as the transformation is not linear as in EGRBI. However, this color space has the advantage of having hue as an vector.

As maize is green, on machine vision based navigation it is important a priori information that can leverage the detection and guide the matching process. But for green colors, there are also quite many different hues. By fine tuning the vector space to give more detailed information regarding the hues of the specific target green, this approach makes it more easy to develop algorithms that can detect the maize by simple thresholding. The improved EGRBI algorithm, ECCI algorithm, was discussed in [9]. ECCI uses a more generic algorithm by adjusting the EC channel to correspond more on the exact color compared to EG in EGRBI, while the cross product channel needs to be still orthogonal. The algorithm is explained well in previous reports, yet there were some not so clear details regarding the scaling of the values so the signed integer channel used in OpenCV code does not overflow. In the ECCI color space, for separating the interesting pixels, all the channels were thresholded into boolean channels and then all channels were bitwise OR-ed into one image that presented the pixels matched.

Also, on top of the color space, the calibration of camera values like exposure and light color affect the result. Too bright or dark image captured by the camera makes it hard to differentiate the hue due to low or high saturation. These anomalies had an impact on the processing regardless of the transformation used as the transformation does not contribute by adding information. The approach to too bright or dark pixels was to threshold these out by intensity or saturation channel depending on the color space.

For actual algorithms, two approaches were developed further as main approaches. ECCI was used to detect the maize while HSV was found good to detect yellow and pink balls. For tuning, sliders were implemented to tune different filters, as well as mouse click on values to identify the details of pixel data regardless if it had 3 or 1 channel in picture that was visualized. The pink in particular was quite far from the hues and was easy to recognize. The yellow was more of a concern as the hue is also close to green in HSV space and if there were old leaves, these could have had a color similar to yellow balls. Luckily in the competition, no yellow colors were present.

As for detection tuning, for ECCI there were parameters for minimum EC, minimum and maximum intensity, and the maximum value cross channel was allowed to vary from zero. For the HSV based method, minimum and maximum values were defined for each HSV channel. While hue provided most information, the rest of the channels were needed to threshold noise, as well as dark and bright pixels.


Original
Perspective
Blockwise
ErodeDilate
EC
Intensity
CR
IntensityTh
D:\Fieldrobot2017\Code\Robot\MV Replay\ReplayMV\Debug\ReplayMV.exe

```
Adding together thresholded images
Results of the line detection script: delta_d 0.027500, delta_theta -3.000000
ECCI transform
Thresholding
Adding together thresholded images
Results of the line detection script: delta_d 0.094583, delta_theta -5.000000
ECCI transform
Thresholding
Adding together thresholded images
Results of the line detection script: delta_d 0.056250, delta_theta -3.000000
ECCI transform
Thresholding
Adding together thresholded images
Results of the line detection script: delta_d 0.017917, delta_theta -1.000000
BlockShowImage: Original clicked x 229 y 147
BlockShowImage: Original image depth 0 channels 3
BlockShowImage: Original color 126 138 148
BlockShowImage: Original average 114 125 136
ECCI transform
Thresholding
Adding together thresholded images
Results of the line detection script: delta_d 0.027500, delta_theta -1.000000
BlockShowImageQuickslime: EC clicked x 203 y 88
BlockShowImageQuickslime: EC value 1854
```

```
//cv::waitKey(0);

//Thresholding. Need to set some
printf("Thresholding\n");
```

Figure 36: Picture of tuning ECCI values and refining the results further into positioning data. Also mouse click on picture was used to get out the details of the pixel raw values.

5.5 Row Detection and Positioning

The row detection algorithm required several image processing operations before the positioning could be done. First the read image was perspective transformed, ECCI transformed and thresholded. Lastly the image was eroded and dilated before the image was passed on for the machine vision positioning algorithm.

After the perspective was fixed, the rows in the picture were straight compared to each other. The image was ECCI filtered and thresholded to detect only maize green as described in detail in previous chapter. Erode dilate process was implemented to filter out noise and strengthen the actual row data points so the result would be as close to four straight lines as possible with minimum error.

A machine vision based positioning algorithm that returned angular and lateral error compared to middle of the row was implemented by calculating histograms. The input image (figure 37)was processed by methods described previously.

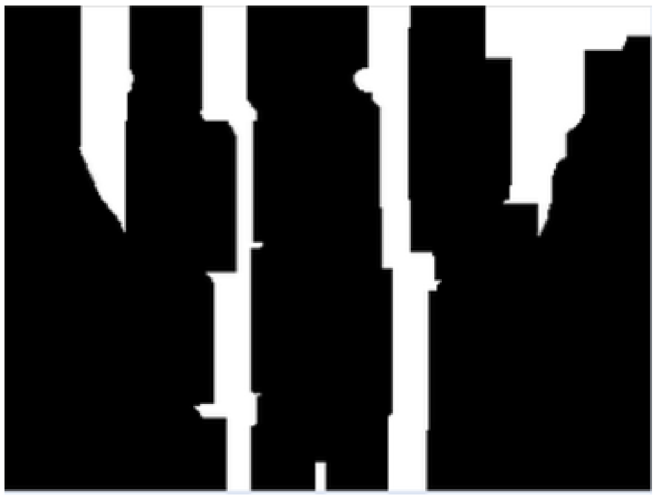


Figure 37: The input image of the machine vision based positioning algorithm. White pixels represent the rows.

The algorithm turned the image with a rotation matrix from -40 to 40 degrees with 2.5 degree resolution and in each rotation the image was divided into four parts based on row width which was priori information. Then histograms were calculated for each part and summed together, then variance was calculated for each histogram. Figure 38 shows an example of histograms of a good rotation and figure 39 shows a bad example. These figures demonstrate that variance of a good rotation large.

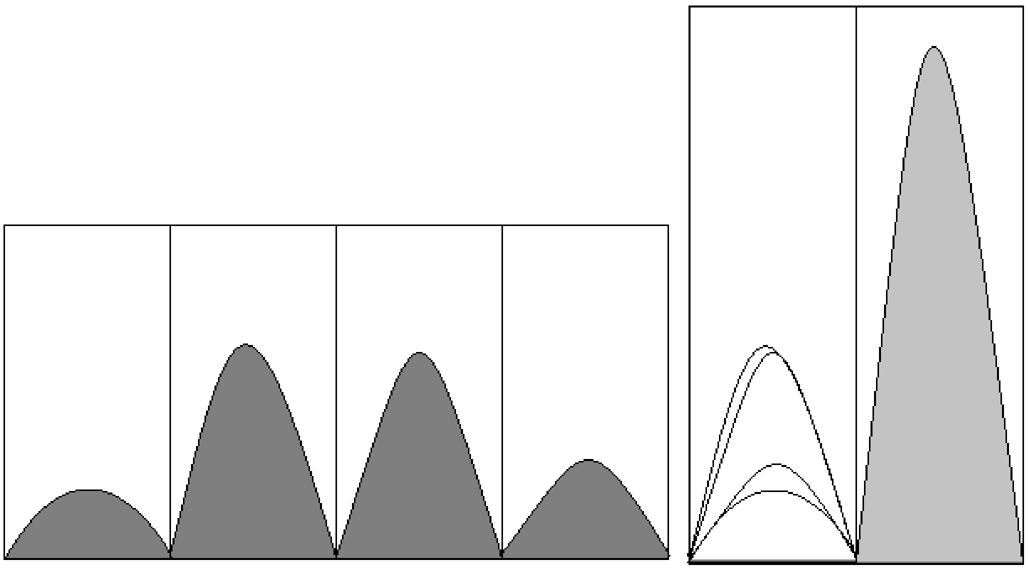


Figure 38: The left figure presents the histogram of the input image (figure 37). It has been divided into four parts based on the row width. The right figure presents the summing of those four parts together.

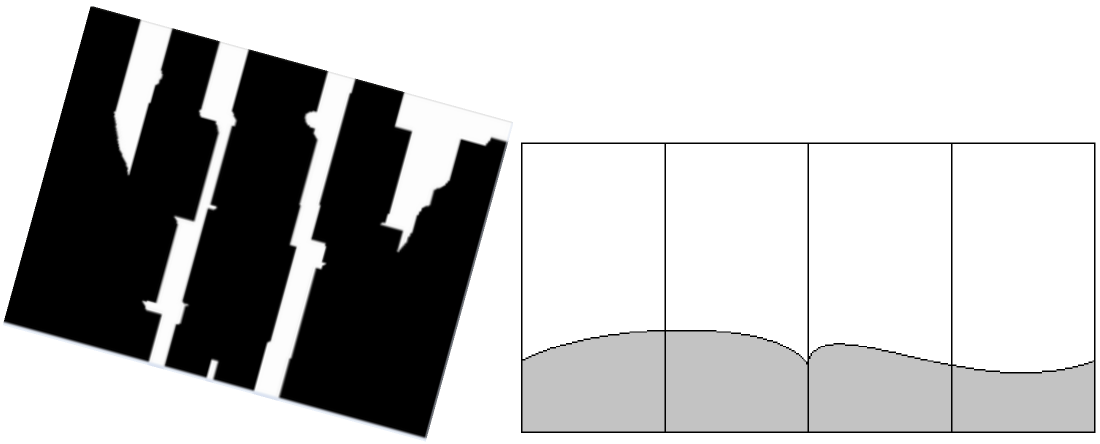


Figure 39: An example of an unoptimal rotation and its histogram. As you can see the variance of the histogram would be close to zero.

The rotation with greatest variance gave us angular error. The lateral error were calculated from that rotations highest peak's offset from the middle of the defined box. Without summing the predefined parts before calculating the variance would give the angular error as easily but the calculation of the lateral error would become much more complicated.

However, despite that machine vision based row detection worked well in simulations, it was so inferior to the laser scanner based row detection in terms of efficiency and accuracy that it was never properly used online. In addition, not using the row detection machine vision algorithm improved the performance of the machine vision program since the algorithm required a lot of processing power. The processing power demands could have been lessened by reducing the complexity of the histogram calculations by choosing a smaller resolution. However, this would decrease the accuracy of the algorithm.

5.6 Weed Detection

In the competition, weeds were presented as pink balls, and these were the point of interest in the machine vision images captured from the cameras. There were two algorithms implemented. First there was an algorithm based on ECCI, but pink color does not set a similar orthogonal vector space as green has as it is more of a mixture of all R, G and B channels. Due to this, another weed detection algorithm was developed based on the HSV algorithm, as in this color space pink has a very distinct hue that allows easy separation of color information.

On top of the color information, for balls, it was possible to consider more criterias for the match. As for the OpenCV algorithm, the blob detection was used to threshold the matched groups of pictures. This would allow also other constraints on the shape. But due to the low resolution used in the cameras (320x240), the golf balls had quite low number of pixels, and initial attempts to use these did not prove feasible so just the number of pixels were used for the matching.

5.7 Tracking

Once the weeds or obstacles are detected on the field, these detection need appropriate actions from the robot, such as an audible signal when the detected object is near the robot. Additionally, the detection has to be identified as either previously known or novelty observation to indicate correct amount of detections. Furthermore, the detections themselves could be false positives in a single scene, which should not be classified as a true positive detection.

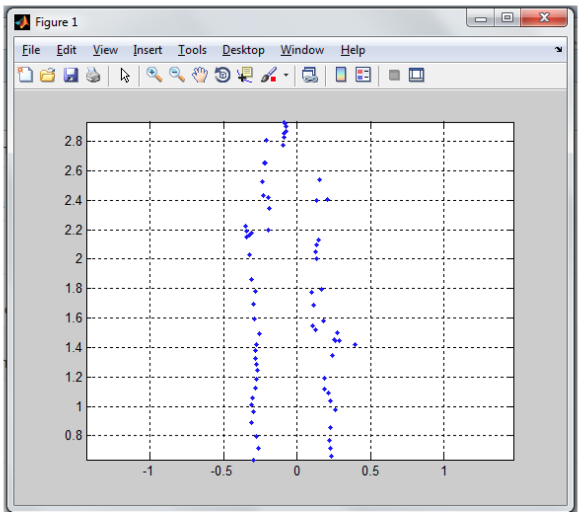


Figure 40. Two separate balls machine vision detections plotted within the visible window of camera, 0.7 to 2.8 meters.

For the clarity of required action, it was necessary to keep track of the detected objects across several images to determine whether and when to indicate the detection of the object. The tracking of the detected objects is done in Simulink with a custom embedded Matlab function. The purpose of tracking is to utilize the incoming detection matrix and the changes in the robot's movement over time steps to determine the uniqueness of the detections by estimating the current location of previously detected objects and comparing the estimate with the real data. This comparison is done using Euclidean distance between the estimated and the real detection position and comparing it to a tunable threshold. If the comparison test is passed by the object, the counter for successfully tracking that object increases. In another hand, if the comparison failed, the counter for unsuccessful tracking increases. These counters are compared against tunable threshold values to provide better decision making when to reliably announce the object detected and when abandon the efforts of tracking it. Additionally, when the previous object is not detected, a new target for the object tracking can be created.

The output of the function is the detection announcement, given at tuned distance value, when used in combination with the beeper. Then, the coordinates of the tracked object could be used for the precision spraying task to spray correctly on the object.

The tracking function can be used similarly with both obstacle and weed detections with different values for the announcement distance of the detection as different actions are required depending on the type of the tracked object. The weeds are announced as late as possible given that the detection accuracy increased closer to the robot. In contrast, the obstacles are announced earlier as the robot will need time to stop and move away from the obstacle without driving over it.

The tunable threshold values are set using the Simulink robot architecture and the log files generated using Log Replay program after tuning the MV algorithm parameters. The tuning was initially done successfully with a responsive Matlab script of the tracking function. However, the integration of this function with the actual Simulink architecture was unsuccessful at the competition site. The indexing of incoming matrix was reversed, hence, objects were only detected when the object was within 10 cm of the camera.

6 Freestyle

For the freestyle task, the team decided to implement a metal detector trailer capable of detecting various metal objects. The trailer was connected to the robot and the robot could autonomously pull and guide it in the field. Otherwise the trailer is autonomous in the sense that upon detecting a metal object it will give light and sound signals independently of the robot. The detection information is also sent to the robot via CAN.

6.1 Background

It is not uncommon that various metal parts end up in the cultivated areas where they do not belong. For example machine parts from heavyweight agricultural machinery, such as hitch pins or balls, may sometimes get lost in the field. Unfortunately, also scrap metal and metal waste is sometimes found among the crop plants. Finding such valuable or even harmful objects, when they end up in the wrong place, is necessary and beneficial. Of course also other use cases are viable for consideration, for example, even an archaeological site might be discovered.

6.2 Functional design

For the freestyle task the team designed, manufactured and assembled a metal detector trailer capable of detecting various metal objects. The design included both mechanical and electronic design, as well as the design and programming of the microcontroller logic.

The functionality of the metal detector is based on an oscillating circuit called the Colpitts oscillator. The Colpitts oscillator is a general type of radio frequency (RF) oscillator circuit that uses a combination of inductors and capacitors to generate an oscillation. The oscillator is tuned by the resonance between the inductor and the combined capacitances of two capacitors in series. [10] Part of the circuit is an inductor, in this case a large coil, that creates a magnetic field once current is applied. Moving the coil across a metal object changes the magnetic permeability in the inductor core. The resulting change in inductance causes the frequency of oscillation in the circuit to change. The change in the frequency can be observed with a microcontroller, for example.

6.3 Prototyping

Before the final design and implementation, some quick prototyping was done as a proof of concept. First, the electronic functionality of an initial metal detecting coil circuit was tested and verified with a prototyping board and an oscilloscope. The prototype was further developed to include some preliminary processing of the oscillating detection signal with a microcontroller. During the prototyping phase some performance tests were conducted. It was observed that the metal detector was unable to detect small objects, such that a commercial metal detector would detect. Unfortunately, the time for design iteration was limited and thus no major modifications were made to the original circuit. The oscillation frequency of the poc circuit was measured around 100 kHz, which might not be optimal for the purpose. Due to time constraints, this was not investigated further.

6.4 Electrical design and manufacturing

The electrical design was implemented according to the original prototype with some necessary additions and modifications. A Crumb128/-CAN V5.1 microcontroller module [13] was used as the processing unit of the circuit. The circuit schematic and layout for manufacturing was designed with KiCad EDA. The circuit board was self-manufactured with basic PCB etching methods and equipment. The components were soldered by hand at the project working facilities.

The final design of the electrical circuit is presented in Figure 41, including the Colpitts oscillator, input-output connections, other necessary components and connections to the MCU. The inductor is a custom made, stranded wire, two-layer, 26 cm diameter coil.

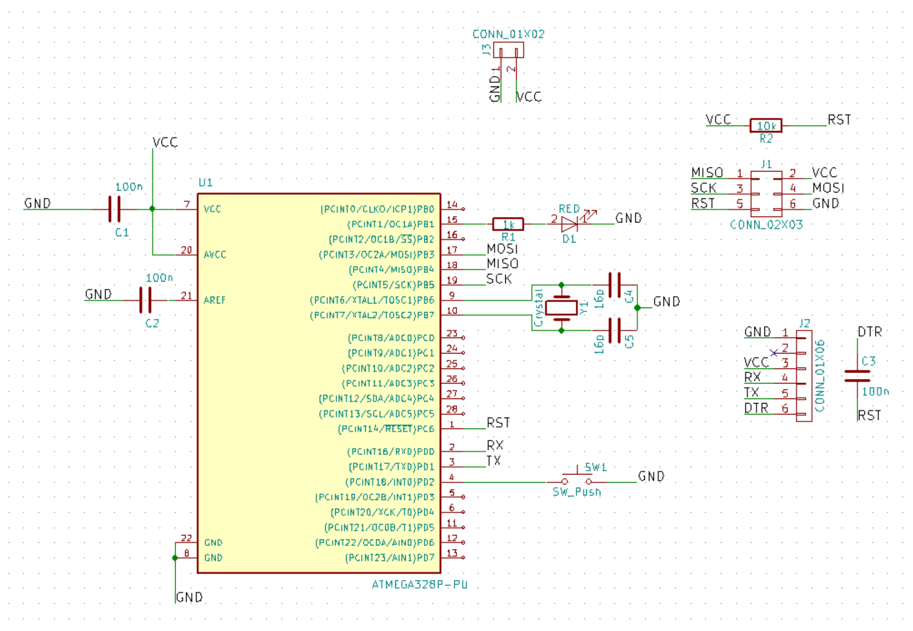


Figure 41: Schematic drawing of the metal detector trailer circuit.

6.5 Mechanical design and manufacturing

The mechanical design was first discussed and then iterated in the 3D design software. The 3D modelling of all individual parts as well as the test assembly of the trailer design was done with PTC *Creo Parametric* 3.0 M050. Apart from the towing beam, all of the parts were manufactured from PLA filament with Ultimaker 3D printers.

A few options were considered with regards to the movement of the trailer. The leading design proposition included two wheels on the opposite ancillary axes of the coil, and this design was eventually implemented. Both wheels consist of two sections and they were directly attached to the axes without any bearings or such. Both wheels are free rolling.

The coil frame was designed to be sufficiently large in diameter to create a larger magnetic field. Due to dimension limitations of the Ultimaker 3D printers the coil frame was designed and made out of four identical parts that were assembled. The design was made such that the coiling could be done in one or multiple layers.

Finally, a casing was designed and manufactured to house the circuit board including the microcontroller, as well as other parts of the necessary electronics. The casing was installed on top of the coil frame, and a speaker was placed on the top of the casing.

The trailer is connected to the robot with a towing beam. The towing beam is connected to the robot rear axle module with a small joint. The towing beam was cut from aluminium and the joint was connected to the beam with a machined aluminium inset. A rendering of the fully assembled 3D CAD model of the trailer can be seen in Figures 42 and 43.

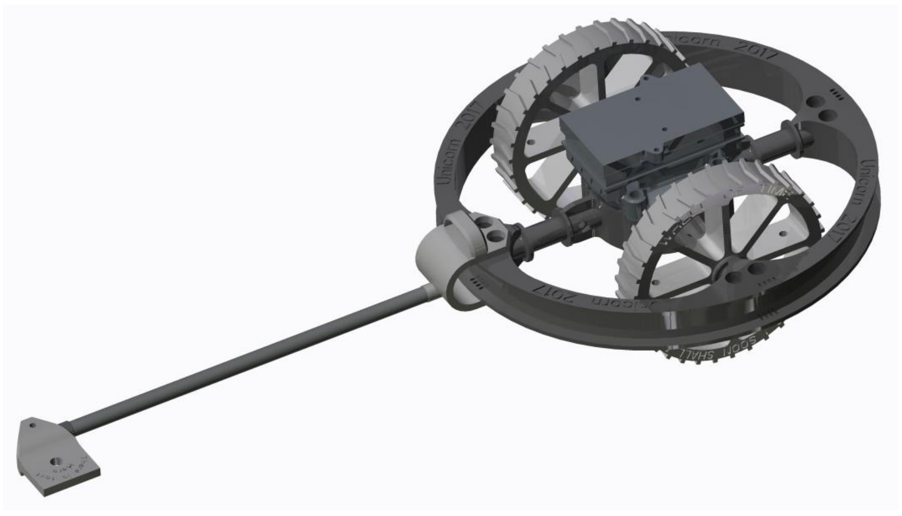


Figure 42: A rendering of the fully assembled 3D CAD model of the metal detector trailer.

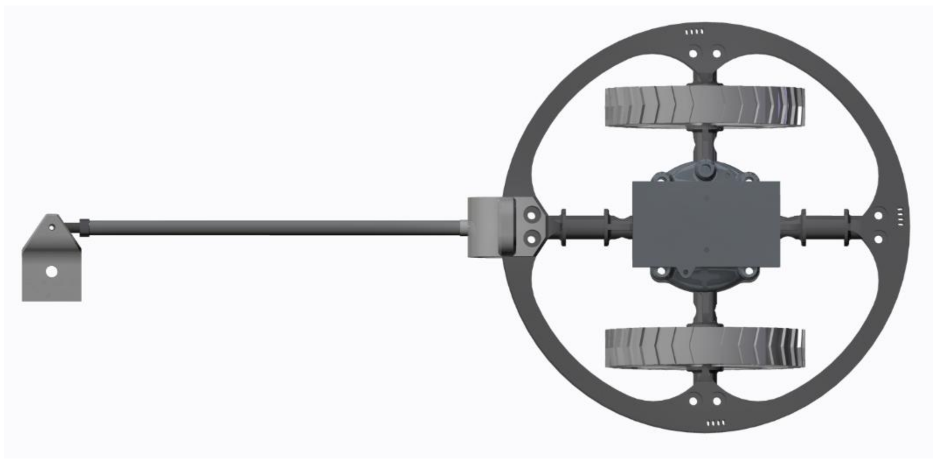


Figure 43: A rendering of the fully assembled 3D CAD model of the metal detector trailer.

An image of the ready made metal detector trailer with the UniCorn is presented in Figure 44. The control circuit is installed in the casing on top of the trailer. The detection indicator speaker was not 3D modelled but it can be seen in the image, as well as the indicator LEDs installed around the circuit board casing.

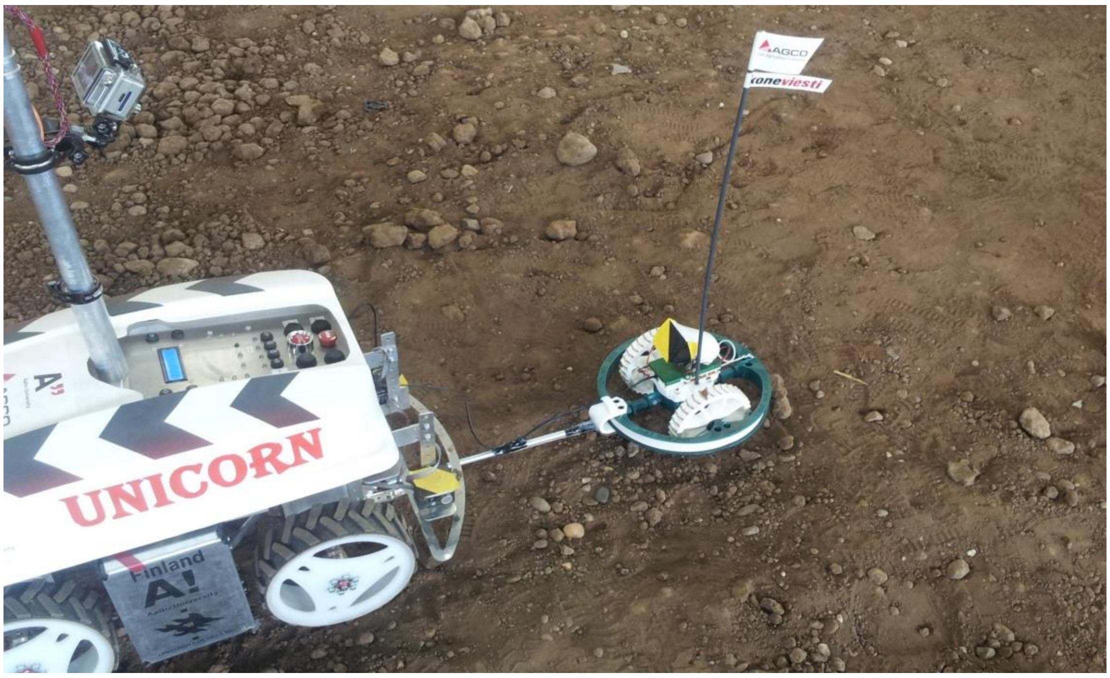


Figure 44: UniCorn and the trailer at Harper Adams soil hall.

6.6 Microcontroller programming and logic

A Crumb128/-CAN V5.1 microcontroller was used to control the circuit. The control logic was programmed with C with CodeVisionAVR. The code runs in a while loop, where the detection is observed from the frequency of the oscillating input signal, the detection data is sent to CAN, and the output signals (LEDs, speaker) are set according to the detection status. The frequency of the input signal is estimated using one of the MCU timer signals. Also a detection sensitivity threshold was set and manually calibrated in the code. In addition, a time threshold was used to verify the detection, i.e. the duration that the detection signal stays above the sensitivity threshold is monitored as well. The duration is tracked by maintaining a timestamp of the last time of detection.

6.7 Performance

The performance of the final product was initially tested manually with various different size and material metal objects. The final performance testing was conducted at the event competition field the day before the freestyle task. Objects with large surface areas were detected very reliably, however, they could not be buried very deep, no more than a few centimeters of soil. Based on earlier tests it was evident that small objects could not be detected. With some additional tuning the metal detector was able to detect for example flattened aluminium cans with high reliability.

Using the trailer with the robot was not without problems, although the robot was able to fully autonomously tow the trailer in the field along a predefined path. For one, some speed constraints with regards to the reliability of the metal detection were observed. With high enough speed the detection of previously detected moderate sized objects became unreliable. Moreover, with the current design, it is not possible to reverse without causing damage to the trailer while the trailer is connected to the robot.

7 Discussion and Conclusions

The UniCorn project involved depth and complexity for applying knowledge and working in a project team, by bringing together multiple engineering disciplines and covering a lot of ground within each field. The project target was a single autonomous robot, but robots such as a field robot are very delicate pieces of hardware.

The project involved a lot of architecture layers on different levels. The hardware assembly as mechanical robot base architecture, defining the kinematics and possible operation modes. On top of the mechanical assembly, there was a computing and communications architecture involving multiple computers and interconnecting busses.

As learning platform with the half year term given, the learning goals were definitely set high. With little prior knowledge, the success of the project depended on utilizing the given platform with robust solutions. The instructors provided the team with hints and training on best approaches to lower the ground, nonetheless, the level of understanding required to master was high. Also, the previous year’s reports provided information regarding the working methods of the previous years.

Not only did the project involve engineering aspects, but team management aspects as well. In order to achieve the goal, the team needed to function and perform well so the tasks could be divided so each team member can contribute. The given schedule required the scalability by team work. For solo work, building the robot within given time would have been next to impossible.

To facilitate the teamwork, the project tasks and integration goals had to be set. In addition, as the architecture began to emerge and solidify, the tasks became more tangible. Yet as an autonomous robot, each subsection of the robot contributed to other and the interfaces between these subsections had to work. To make this happen, parties involved were required to understand the underlying terminology and principles. The documentation and communication played a big role to make this happen.

As for the subsections, members and scalability, the project had smaller subprojects using the members as resources. One difficult project management and personal aspect was how to balance the workload and prioritize the different tasks, so bottlenecks and dependency problems would be as small as possible. This was made more challenging by the difficulty to estimate the workload of each task. As the team members lacked experience with this kind of work, feedback was asked from the instructor and the advisor. Nevertheless, estimates of the time needed for tasks, especially the advanced navigation, were underestimated. This was due to the complexity of the tasks as well as unforeseen setbacks, such as persistent bugs. Time management, prioritizing and handling setbacks and uncertainty are among the valuable lessons learned from this project.

For the UniCorn specifically, the most challenging aspects creating trouble for the development were related to communications, architecture changes and keeping related interface changes managed. The project utilized revision control for changes, but using this together with Matlab and Simulink emphasized the communications and documentation as these products provided no transparency to track differences in code. Only hints of the changes could be seen from comments given during the code commits.

For the interfaces, not only the concrete software or hardware interfaces matter, but also the established best practices and conventions. As the robot integration required integration between software components, it was essential to keep the other parties updated on the current situation. On this part there were also some communication challenges and misunderstandings that contributed to generate extra work.

An example of such were parameters that change the code behaviour between simulation mode and online mode. Also, problems arose from the practise and understanding which files are kept in revision control.

The project goal was the Field Robot Event competition and UniCorn with 7 out of 8 team members attended the challenge. The team was well prepared with the robot navigation, but due to late integration and aspects during the final in rush integration, the machine vision and sprayer were not brought to operations. At late stage of the project when the academic term was normally over, the project continued but there was still parts that required work to be finished and this brought pressure on the members present. There was a team decision involved on prioritizing the navigation, as it laid the base ground for the more advanced functionality. However, given the sparse resources available with the pressure given during the final stages of the project, the time grew short and the final integration of machine vision functionality was given very narrow chances to be functional even as the subcomponents worked adequately well. The final integration work was postponed all the way to the actual competition field, and done on field. This also had negative impact on the work, due to the success pressure and the competition field conditions not making the work any more easy.

Overall, the team was very focused on the project and the overall contribution of members towards the goal was impressive. For everyone, this was one of the most time consuming courses during the years at Aalto University. Yet, this gave more complete package of information how to approach and what it involves to complete a highly complex project.

This project would not have been possible without our sponsors from AGCO, Valtra, Koneviesti, Ford, Laserle, and SICK.

References

1. *Field robot event*, www.fieldrobot.com.
2. *Agronaut*, http://autsys.aalto.fi/en/attach/FieldRobot2016/FinalReport_Agronaut_Fieldrobot_2016.pdf, Field Robot Event 2016, p. 5-11, 18.
3. *Threaded fasteners*, https://www.wurth.ie/media/downloads/pdf/short_form_catalogues/Threaded_Fastners.pdf , Würth, p. 44.
4. *Dimensioning metric screw assemblies*, https://www.wuerth-industrie.com/web/media/en/pictures/wuerthindustrie/technikportal/geschuetzer_bereich/Kapitel_06_DINO_techn_Teil.pdf.
5. Cornivore, http://autsys.aalto.fi/en/attach/FieldRobot2011/Cornivore_FinalReport.pdf, Field Robot Event 2011, p. 23-24.
6. *SICK*, sick.com.
7. DoubleTrouble, http://autsys.aalto.fi/en/attach/FieldRobot2013/DoubleTrouble_FinalReport.pdf, Field Robot Event 2013, p. 21.
8. *CAN Messages*, https://www.kvaser.com/can-protocol-tutorial/, Kvaser.
9. WheelsOfCorntune, http://autsys.aalto.fi/en/attach/FieldRobot2007/FRE2007_WheelsOfCorntune.pdf, Field Robot Event 2007, p. 4 -5.
10. Carr, Joe (2002). *RF Components and Circuits*. US: Newnes. pp. 126-127. ISBN: 0750648449.
11. *GroundBreaker,* http://autsys.aalto.fi/en/attach/FieldRobot2015/FinalReport_GroundBreaker.pdf , Field Robot Event 2015, p.15-17.
12. Dynamixel AX-12A User Guide, p.7-18 http://www.hizook.com/files/users/3/AX-12_Robotis_Dynamixel_Servo_UserGuide.pdf
13. Crumb128/-CAN Datasheet, http://download.chip45.com/Crumb128-CAN_V5.0-5.1_infosheet.pdf.

VOLTAN

Velázquez López Noé[1] Reyes Amador Armando[2], Ruiz González Luis Gerardo[2], Sánchez Chávez Iván[2], Garrido Cuapantecatl Norberto[2],Lemus Reyes Swamy Yatzar[3], , Morales de Jesús Cesar[3], Pilón Alcalá Cesar Augusto[3],

1. Universidad Autónoma Chapingo. Departamento de Irrigación.
2. Universidad Autónoma Chapingo. Posgrado de Ingeniería Agrícola y Uso Integral del Agua.
3. Universidad Autónoma Chapingo. Departamento de Ingeniería Mecánica Agrícola.

INTRODUCTION

Field Robot Event (FRE) is an annual competition for agricultural field robots. This year, 2017, the 15th FRE was held at Harper Adams University in England.

Our robot was built from scratch to meet all requirements not only for the competition but in general for agricultural activities. It was the first time that our University (Chapingo Autonomous University) and a Mexican team participated in this competition and everything was very challenging.

One of the main achievements for this competition was the autonomous control and guidance of the vehicle between the maize rows just using computer vision.

The team would like to take the opportunity to thank CLASS foundation for their support this year.

Design.

In agricultural machinery the speed is around 6km/h. That was one of the parameters to design the transmission and to choose the motors of this robot. As for the total weight of the robot 60kg were considered including some implements that are attached to its frame. The size of the robot are 60 cm wide, 70 cm long and 50 cm high.

Manufacturing and selection of components.

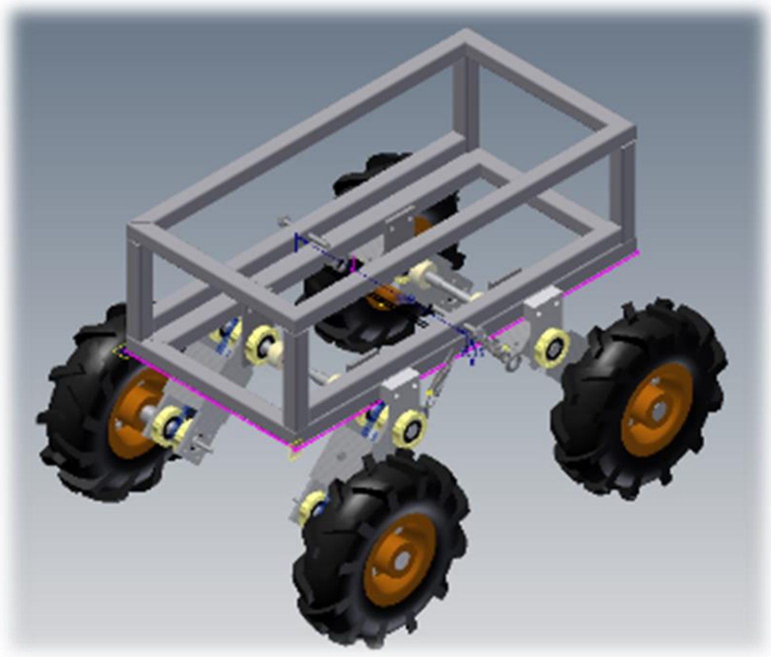


Figure 1. CAD draft of frame of the robot.

The selection of mechanical elements was carried out based on calculated and desired parameters of the vehicle.

Tires.

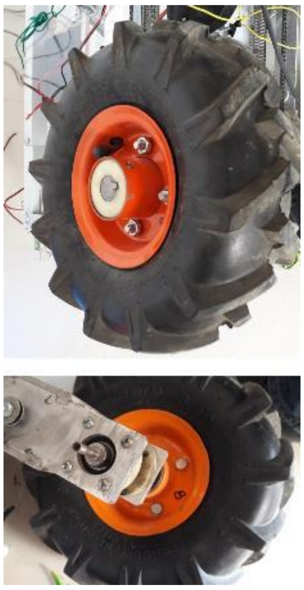


Figure 2. Tires used in the vehicle.

To have good traction on any kind of soil conditions, agricultural tires were selected. These allowed for stability in the field.

Power Support Arms.

The wheels support arms were manufactured of aluminum to keep the vehicle as light as possible (Figure 3). These arms allow for adjustment of the clearance height of the vehicle. The bearings were adapted by placing bushings made of nylamid,

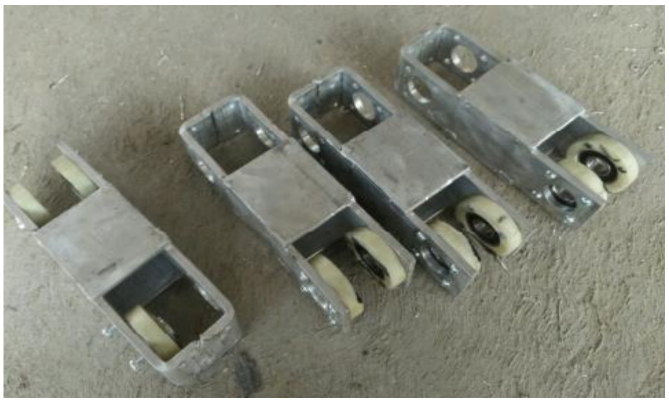


Figure 3. Arms, bush and support of the tires.

As for the assambling the wheels to the main frame shaft were manufatured(figure 4), which were used also for trasmiting power with a chaing and a couple of sprockets..

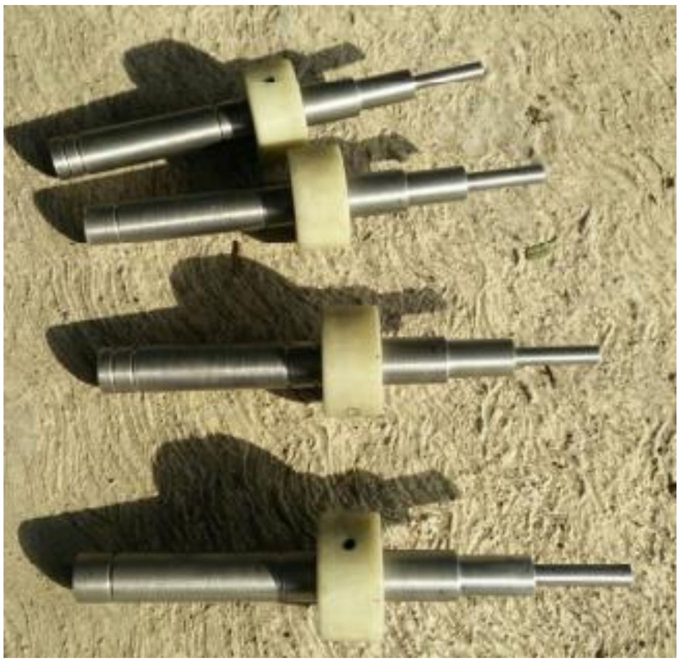


Figure 4 . Shaft and bush.

Electric motors and control

Due to its low cost and power capabilities a couple of electric motors used in Power Wheels toys vehicles (figure 5) were selected, these motors torque and transmission, which meets our speeds requirements and power requirements.

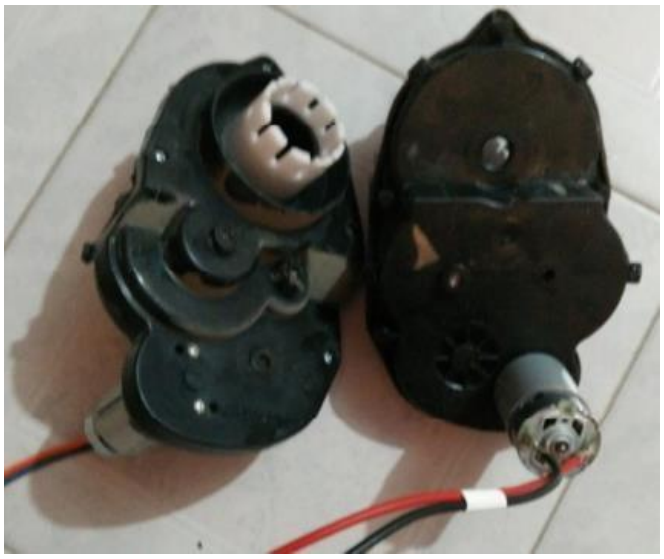


Figure 5. Electrical motors used in the vehicle.

To control the motors two drivers IBT-3 were used. This driver is designed to control speed (PWM) and turning direction and meets the requirements of the current consumption of the motors (2-50A)

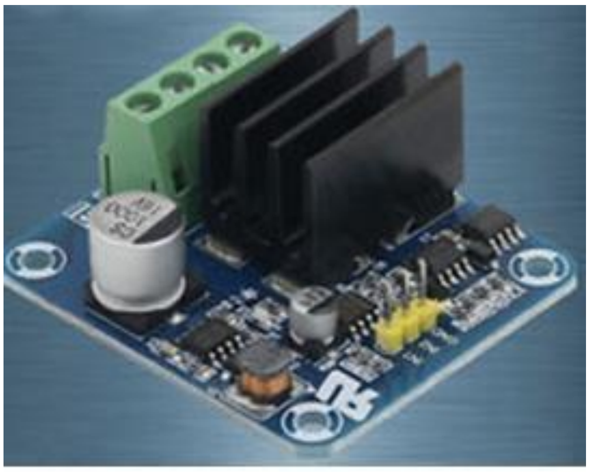


Figure 7. Driver IBT-3.

Arduino mega arduino (2560) was selected for its versatility to carry out projects and in particular for having exclusive pins for PWM ideal for speed control of motors for this application (figure 6).

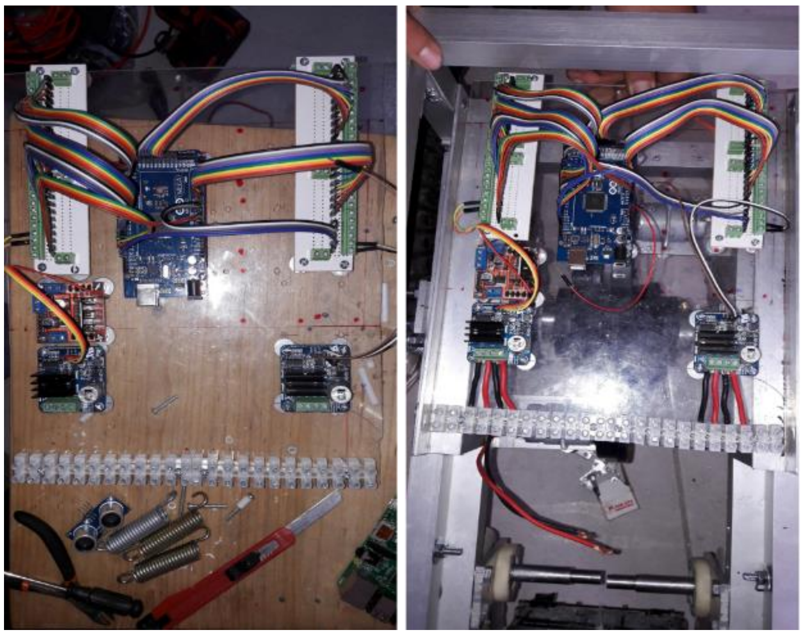


Figure 6. Assembling of the main board to control of the vehicle.

Batteries.

Two 12V / 12A / h batteries were installed in parallel. Both were install in the center the vehicle to keep the center of mass balanced.

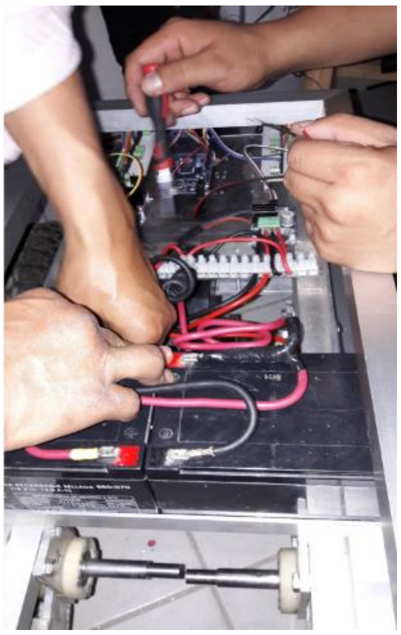


Figure 8. Batteries used in the vehicle.

Encoder EE-SX1103

As a sensitive element of the vehicle it was decided to use an optical encode to track the vehicle trajectory and positioning.

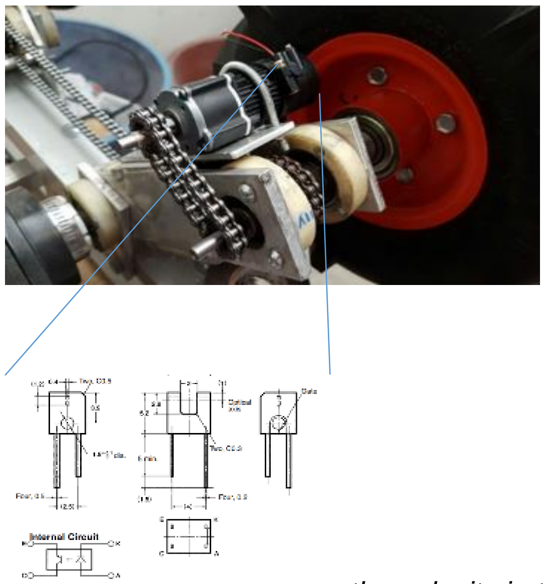


Figure 9. Encode used to control the velocity in the vehicle.

Frame (structure).

Finally all the above was mounted to the main structure, the most important modification was to reduce the size, making it 0.29 m in height, for that the vertical bars were then cut and then re-joined to the bases, this replacing the rivets, which was made with a drill, and using the cutting blade. At the end it was reconnected with screws and nuts, to allow more practical disassembly in case it is required for future modifications.

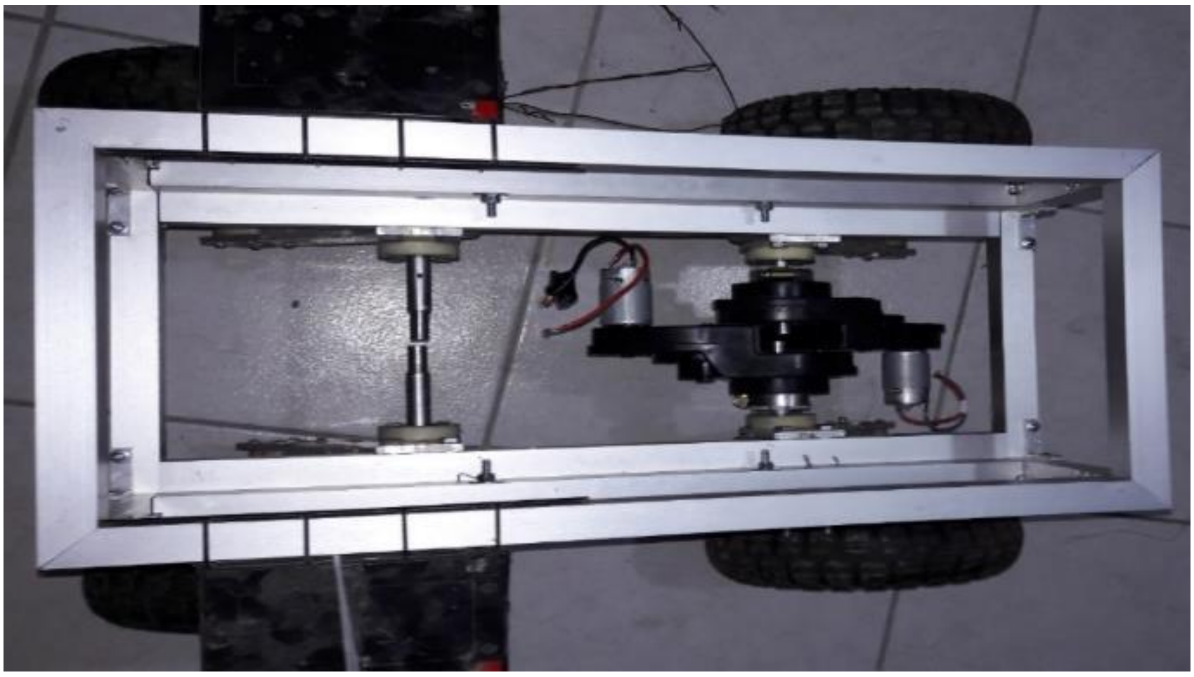
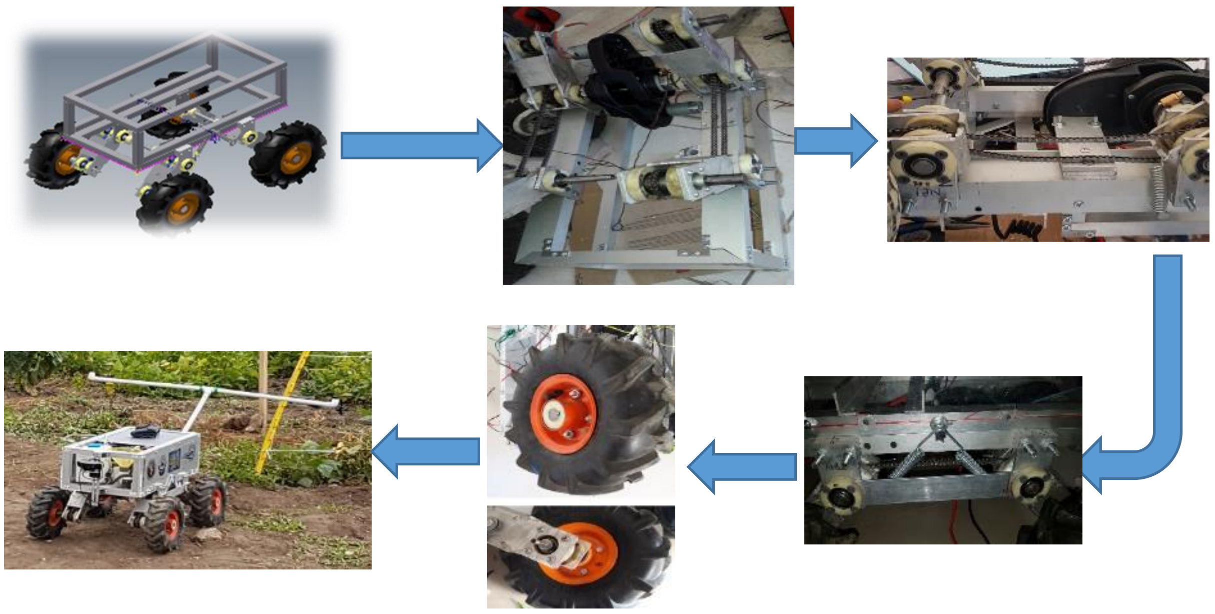


Figure 10. Main frame (structure) of the vehicle.

The main elements that make up the prototype are listed below. ()

- Frame.
- Power unit.
- Electrical systems.
- Transmission systems.
- Tires.
- Attachments.

Figure 11 Manufacturing process



Ultrasonic sensors

he ultrasonic sensor (figure 10) were used to detect the end of the rows and were selected due to its low cost.

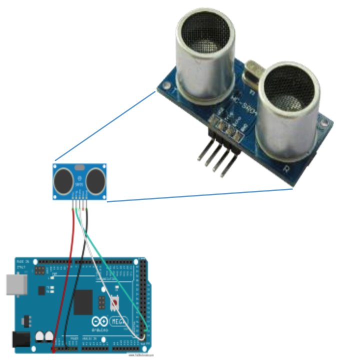


Figure 12 HC-SR04 ultrasonic sensor installed in the vehicle

Computer and Vision algorithm

We are using a Dell notebook with intel core I7 processor, 16 GB RAM and 250GB Hard disk. An algorithm was developed in visual studio with OpenCV for the autonomous navigation

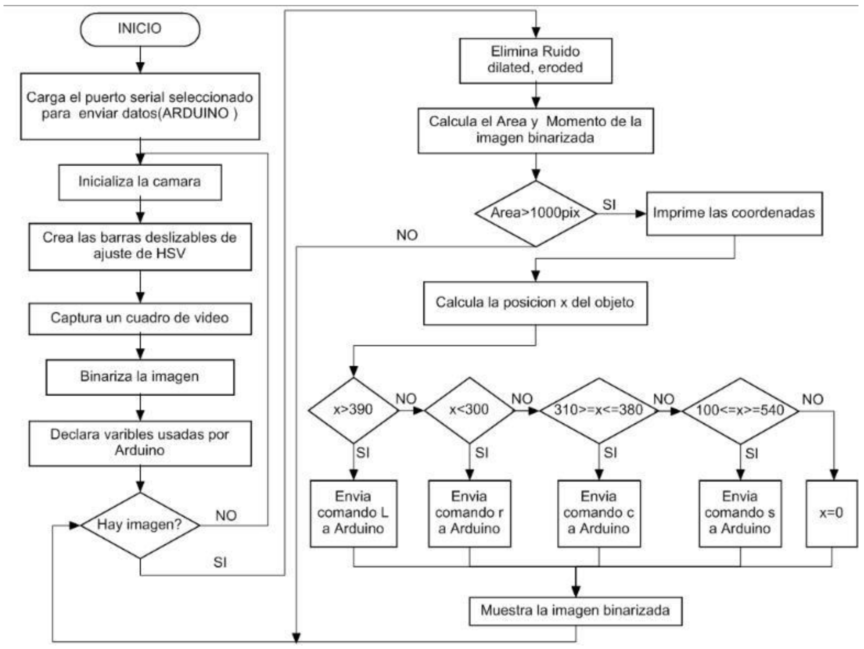

INICIO
Carga el puerto serial seleccionado para enviar datos(ARDUINO)
Inicializa la camara
Crea las barras deslizables de ajuste de HSV
Captura un cuadro de video
Binariza la imagen
Declara varibles usadas por Arduino
Hay imagen?
NO
SI
Elimina Ruido dilated, eroded
Calcula el Area y Momento de la imagen binarizada
Area>1000pix
SI
Imprime las coordenadas
NO
Calcula la posicion x del objeto
x>390
NO
x<300
NO
310>=x<=380
NO
100<=x>=540
NO
SI
SI
SI
SI
Envia comando L a Arduino
Envia comando r a Arduino
Envia comando c a Arduino
Envia comando s a Arduino
x=0
Muestra la imagen binarizada

Figure 13 Autonomous navigation algorithm

The algorithm consists of detecting the two rows of maize that are aside the vehicle. Then we calculated the center of mass of these two rows which happened to be the exact center of the row. Finally, through serial communication the main computer sends signals to the Arduino for motor control (Figure 14)

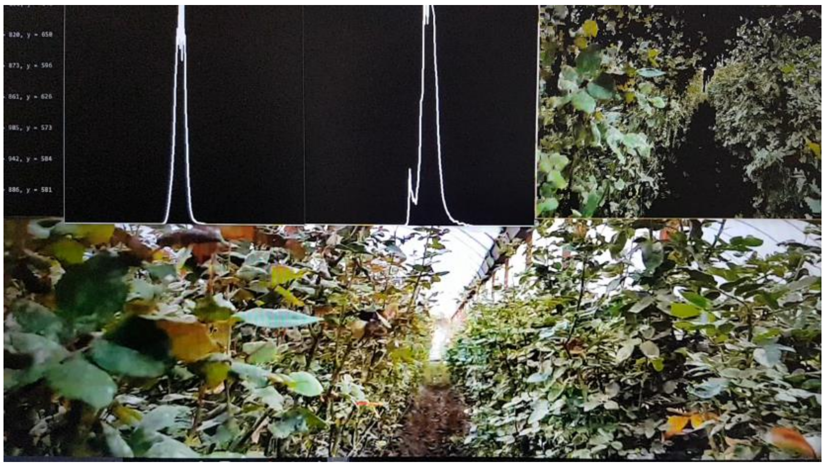
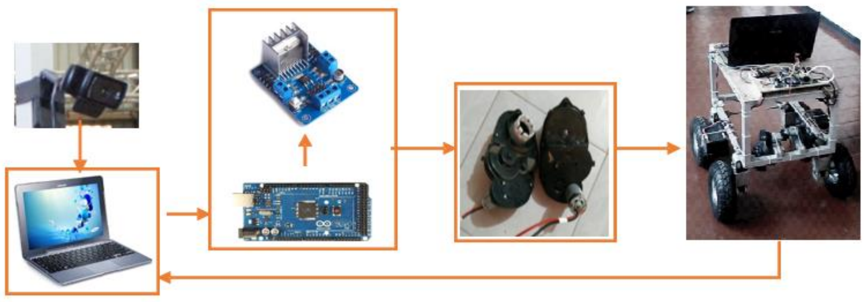


Figure 14 Developed vision system.

Controller Architecture



CONCLUSION

Our robot competed for the first time in the Field Robot Event 2017. This robot was designing to be multitasking, so many applications can be added. Our project started 3 years ago and this year we finally could compete. The main achievement this year was the computer vision autonomous navigation, which also allows for weeds detection on the field.

REFERENCES

The construction of the robot was supported by Chapingo Autonomous University and CLASS Foundation

Zukbot

Name of Institution	Gdańsk University of Technology
Department	Department of Automatic Control
Country	Poland
City	Gdańsk
Webpage	http://fb.com/zukbot
Team members	Błażej Błaszczuk, Kamil Domański, Małgorzata Kraszewska, Mateusz Olszewski (C), Patryk Reiss, Adam Strużyński, Katarzyna Studzińska, Tomasz Węsierski
Instructors	Stanisław Raczyński
Contact Email	mateusz.olszewski@skalppg.pl

Weight (kg)	15
Actuators/motors installed	4x BLDC Motors
Turning radius (cm)	0
Battery voltage	25,2v
Battery duration (mins)	45
(W x L x H) (cm)	50 x 65 x 50
Sensors installed	SICK TIM551, SJCAM 4000

Robot software description
Robot runs on ROS(Robot Operating System) Installed on Linux Ubuntu. For image recognition purposes OpenCV library was used
Robot hardware description
Robot is controlled by Gigabyte Brix i7 miniPC. PC communicates with embedded peripheral drivers through Ethernet interface. Peripheral drivers are based on STM32F4 microcontrollers. Every peripheral board implements ethernet. Data received from PC is sent through CAN interface to motor drivers. CAN interface is the best choice for control of high power actuators due to insusceptibility to electromagnetic interference thanks to differential transmission line. BLDC motors are driven by VESC open source brushless motor controllers using field oriented commutation. Robot is powered from 6 cell Li-Po battery with built-in battery management circuit. The robots rigid construction is based on aluminium profiles and plates, that ensures robot stability, as well as easy installation of sensors and weeding module. High power from brushless motors have been transferred by synchronous belts with 4:1 RMP ratio. Used wheels are from big RC models. Many parts of this construction was printed on a 3D printer. The robot is fully waterproof.
Task strategy description
Field navigation is based on readings from laser scanner, with if then fused with IMU(Inertial Measurement Unit) data to form a map representing position of every plant relative to the field. On this map are also placed markers found by image recognition.

Team Photos

Robatic Group
Wageningen University &
Research

Robot: Agrifac Bullseye

Kamaro Engineering
Karlsruher Institute of
Technology

Robot: Beteigeuze

Schulerforschungszentrum
Sudwurttemberg

Robot: Carbonite

Fontys Venlo
Fontys University of Applied Sciences

Robot: Dora the Explorer

Floribot 2017
Heilbronn University

Robot: Floribot

Feldschmiede
University of Hohenheim

Robot: Hefty

FREDT
Technische Universität
Braunschweig

Robot: Helios

On a shoestring budget
Fontys University of Applied
Sciences

Robot: K9

TAFR Team
University of Ljubljana

Robot: TAFR

Moops
Harper Adams University

Robot: Terra

Cornholio
Hochschule Osnabrück

Robot: The Great Cornholio

Team UniCorn
Aalto University

Robot: UniCorn

UACH – Dimabot 17
Universidad Autónoma
Chapingo

Robot: Voltan

SKALP
Gdansk University of
Technology

Robot: Zukbot

Overall Winners Presentation Photographs

Field Robot Event 2017

1st

Kamaro Engineering

2nd

Carbonite

3rd

Floribot 2017

With Thanks to our sponsors:

EurAgEng

syngenta

CLAAS Stiftung

YARA

Knowledge grows