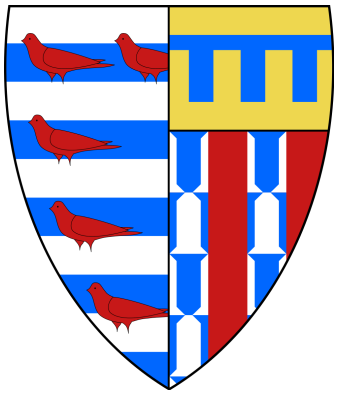
Part IB Integrated Design Project

Engineering Assessment of the Project



Xiaoding Lu

Team 2

Pembroke College

I declare that this assignment is my own work and that I have correctly acknowledged the work of others.

Word count: 2317 excluding appendix

Lab Group 102

Feburary 17, 2018

Contents

1 Introduction **1**

2 Team Management Review **1**
2.1 Overview . . . 1
2.2 Software Sub-team Management . . . 2

3 Design Decisions and Changes **2**
3.1 Choice of Pick Up Mechanism . . . 2
3.2 Choice of Running from Workstation . . . 2
3.3 Line Sensor Position and Line Following Algorithm . . . 3
3.4 Usage of IR Circuit Breaker . . . 3
3.5 Usage of Side Distance Sensor . . . 3
3.6 Plant Detection Algorithm . . . 3
3.7 Discarding Front Distance Sensor . . . 4
3.8 Discarding Counter Circuit . . . 4
3.9 Discarding Plant Collection on Hill . . . 5
3.10 Plant Collection Box . . . 5

4 Prototype Production Cost and Scaling **5**

5 Appendix **6**
5.1 Costing Summary . . . 6
5.2 Unit Cost for Mass Production . . . 6

List of Figures

1 Completed AGV . . . 1
2 Pick Up Mechanism . . . 2
3 Costing Summary . . . 6
4 Unit Cost . . . 6

1 Introduction

Autonomous Guide Vehicles (AGV) are widely used within the industry, being responsible for tasks ranging from repetitive which incurs high labour cost to dangerous which could potentially harm a human operator. The Integrated Design Project (IDP) aims to provide a demonstration on the usages of AGV within the frame of agriculture. Selecting, harvesting and dropping agriculture products is a tedious process and should be made autonomous, reducing labour costs. It also provides challenging software problems such as autonomous vehicle driving and image analysis.
Within the 4 weeks time constraint, a fully working AGV which performs plant detection, collection and drop-off was produced. Figure 1 shows the completed AGV. The software team was able to implement the line-following, hill-climbing, plant(image) analysis, harvesting algorithms through embedded programming.

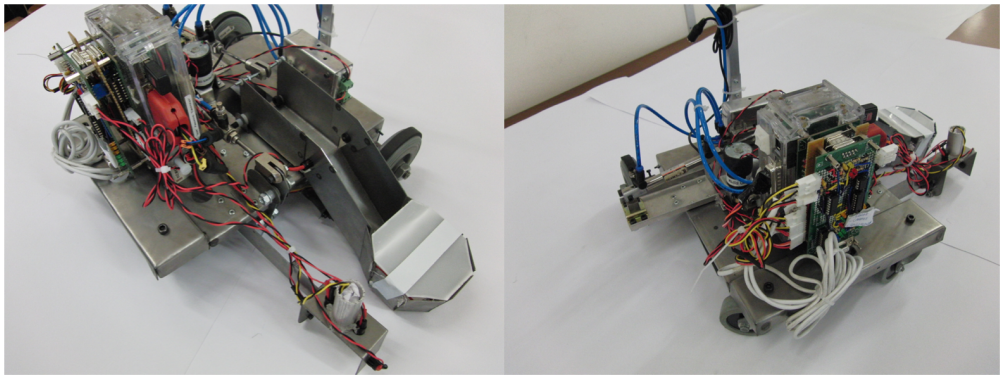


Photo from http://www3.eng.cam.ac.uk/DesignOffice/idp/idp_photos/201811/index.html

Figure 1: Completed AGV

2 Team Management Review

2.1 Overview

Overall the team has managed to produce a fully working AGV within the given 4 weeks time constraint, the result to some degree reflect good coordination between sub-teams.
Slack is used for team members to communicate with each other, three channels are created to accommodate the communication between each sub-team. Overall members of the team show good initial engagement on Slack, with daily communications and updates. Between week 3 and 4, the level of engagement on Slack decreases as each sub-team becomes more focused on their current tasks without updating members of other sub-teams. The issue of communication is more server between the mechanical and software sub-teams as they work in completely different locations.
As a result of this, software code underwent major changes due to uninformed design decisions. Future projects shall place greater emphasis on sub-team communications with either:

i One team meeting per two days

ii One sub-team summary placed in the shared directory per week

iii Daily update on sub-team progress through Slack

Overall, the electrical team has managed to follow the Gantt chart produced at the beginning of the project, the mechanical team deviated from the Gantt chart due to the slight delay in design acceptance, as a result of this, the software sub-team had minimum work in week 2 and first half of week 3, the work load increases dramatically between week 3 and the day of competition.

2.2 Software Sub-team Management

Overall, coordination within the software sub-team has enabled a fully functioning code to be produced on the day of competition, with minimum numbers of unanticipated bugs (only a delay is added on the day of competition to ensure correct navigation of AGV).
However, due to 1. the short time constraint on the project and 2. sub-team members' reluctance in using version control; the majority of the software was coded by one person (out of a sub-team of three), with one person in the sub-team providing minimum contribution towards the code.
Again, this problem could have been mitigated with good communication within sub-teams, but a version control is almost essential had the project been larger.
Furthermore, as the mechanical sub-team only provided a working chassis towards the end of week 3, the majority of the code before week 3 was written under the hypothesis of the AGV's behaviour. The poor communication between mechanical and software sub-team resulted in a major design change in the plant detection algorithm (see section 3). The process of re-writing the entirety of a major algorithm can be risky and stressful, further projects should emphasis on the mechanical team providing software a test chassis with sensors mounted for algorithm testing.

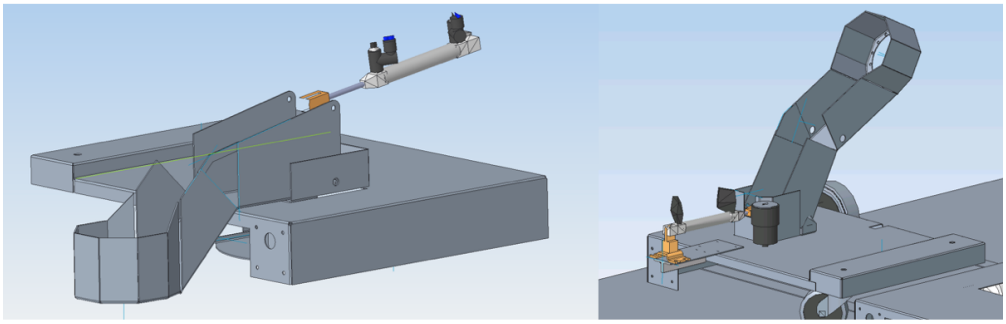
3 Design Decisions and Changes

3.1 Choice of Pick Up Mechanism

The design decision on pick up mechanism was discussed by the team, figure 2 shows the final design decision modeled using Creo. Several other mechanisms were considered, they were seen implemented by other teams in their final vehicle design.
The reason for choosing the rubber band pick up mechanism is that:

i Very simple to implement as there are only two states for the pneumatic actuator to be in.

ii In case of an error in plant-recognition algorithm i.e. incorrectly identifying the size of the plant, the pick-up mechanism will not pick up undersized plant due to rubber band hole size.



Created by Rory Dyer and Charlie Spicer using Creo

Figure 2: Pick Up Mechanism

3.2 Choice of Running from Workstation

From the tasks given in week 1, after comparing the run time difference between running the program from the workstation (instructions sent through wifi) and directly through the on board microprocessor, the software sub-team decided to run the program through the workstation since:

i Not much data is required to be transmitted through wifi

ii In order for the code to be compiled and ran on the microprocessor, terminal must be used which can become tedious

The result of this decision did not affect the performance of AGV, on the day of competition, there is no noticeable difference between teams that run from local workstation and microprocessor.

3.3 Line Sensor Position and Line Following Algorithm

The line following algorithm was coded and marked complete within week 2, however during week 3 the mechanical sub-team noticed a fault in the measurement in the pick-up arm mechanism, resulting in a 3cm difference in distance between the centre of the plant. As a result, the line sensor positions had to be off-centred to the right by 3cm.
Fortunately, due the the way the line following algorithm was originally implemented (with offset between two motors), the basic straight line following works after the changes. However, all the turning were changed due to the fact that the centre of rotation of the sensors is no longer the centre of rotation of the AGV. In 180 turn code, a `forward_till_duration` is coded to perform the turn, this function is later called by docking hence it is not unitized only once in the project.
As a result of the change of configuration of the line sensor position and algorithm, the AGV behaved normally as expected, driving straight with a 3cm offset from the centre of the robot.

3.4 Usage of IR Circuit Breaker

The IR Circuit Breaker was added to the plant pick up arm in week 3, this addition greatly simplifies the plant detection algorithm (see section 3.6). Prior to the addition of the circuit breaker, the plant detection algorithm is able to detect the beginning of a plant and start to push LDR readings to a vector. However, occasionally the algorithm fails to identify the end of a plant resulting in it merging the profiles of two or more plants.
As a result of adding IR Circuit Breaker, whether a plant has been "seen" by the AGV can be determined rapidly and accurately. This reduces both the space and time complexity of the plant detection algorithm and make the code more robust and reliable.

3.5 Usage of Side Distance Sensor

The addition of the side distance sensor, in line with the plant detection arm was done the day before the competition. As the LDR sensor is in front of the pick up mechanism, the AGV scans a plant, determines its type and pushes it to a memory location. The actual pick up action is only performed when the LDR starts to pick up readings for the next plant, it then picks up the plant previously seen.
The original code (without the side distance sensor), albeit able to detect all types of the plants, could not perform picking up if:

i The last plant on the "plantation" is a large plant which needs to be picked up

ii The bean plant was not able to break the distance sensor

Also the pick up mechanism may very rarely miss-align with plant causing the pneumatic actuator damaging the AGV.

3.6 Plant Detection Algorithm

The plant detection algorithm has underwent many design changes. The uncertainty in the under-laying algorithm is a result the fact the LDR sensor arms were not provided until week 3.
Before proper sensor arms are made, a very basic mechanical set up is used to take LDR readings. Initially the algorithm the team came up with involves taking the mean and variance of each plant. After directly implementing the code in C++, the result was disappointing (with only 50% success rate).

Second attempt is made with much more thoughts. The LDR readings are taken and a separate function exports all the LDR readings to an external text file. A python script is written which reads the text file and plots the array using `matplotlib`. The results are investigated and an algorithm that takes the maximum value of each plant array is written and tested to be working with above 70% success rate.
In week 3, two major factors resulted in another complete change in the plant detection algorithm. The electrical team used a potentiometer instead of a fixed resistor value making the LDR more sensitive and provides more stable readings. Also a complete sensor arm was made with direct LED reflections. As a result of this, only the mean of the plant and the size of the plant array was required to determine the type of the plant, this rendered success rate at above 90%. The algorithm is implemented as three private methods in the robot class, namely `plant_forward()`, `plant_pickup()` and `analyze_plant()`, called in the public method plant_loop() which runs the algorithm until a node has been reached. Part of the code for `analyze_plant()` is shown below.

```
void analyse_plant(){
mean = accumulate(plant_array.begin(), plant_array.end(),
0.0/ plant_array.size());
plant_size = plant_array.size();
mean = mean/plant_size;
int port1 = get_input(READ_PORT_1);
if (plant_size<10){
        destroy();
        cout << "Noise" << endl;
        return;
        }
else if (plant_size <= SizeThreshold && mean > TypeThreshold ){
        cout << "Undersized␣Cabbage" << endl;
        rlink.command(WRITE_PORT_1, (port1 & 0x8F) | 0x40);
        }
        ....
mean=0;
plant_array.clear();
plant_size = 0;
```

3.7 Discarding Front Distance Sensor

The electrical sub-team has produced a fully working front distance sensor which is supposed to be used for dropping off plants. The software sub-team decided to not to use these distance sensors, the function docking is written taking three time as inputs, the AGV then performs a mixture of maneuvers to achieve plant drop off.
The decision is made since even with the distance sensors enabled, the drop off sequence will still require thresholds. The result of the design decision did not affect AGV's performance in any way. However if the course was to change then a more general function which utilizes the distance sensor will be more robust and useful.

3.8 Discarding Counter Circuit

The electrical sub-team has produced a counter circuit that is able to provide a signal when the IR circuit breaker mounted on the arm has been broken four times. This is an amazing piece of hardware and it was a shame that it was not included in the software.
Obviously, counting the number of plants collected can be quiet simply implemented by adding a member in the robot class which increments on collection of the plant. This is reliable as long as plants are identified

and collected correctly. Had the algorithm for plant detection been less reliable, the counter circuit would have been used to ensure the correct number of plants are collected within each collection box.

3.9 Discarding Plant Collection on Hill

There has been much debate on whether the AGV should attempt plant collection on hill, as a result of this, the pick-up is mounted on the opposite side of the conventional direction, hence the AGV must perform a 180 degrees turn prior to plant collection on flat ground.
Methods such as `turn_45()` and `forward_till_45()` have been written which handles hill plant collection, the day before the competition, due to both unreliable sensor performance on hill and slipping of the wheels during collection, the software team made a decision not to collect plants on the hill.
As a result of this, four out of six large plants were correctly collected and delivered on the day of competition. The team has managed to win the competition by only making perfect collection and drop off on flat ground. Marks would have been deducted had the AGV attempted the hill and failed to navigate through the course.

3.10 Plant Collection Box

The addition of slanted cardboard within the plant collection box seems to be the most trivial design change. However this was probably the most important design change after the chassis was completed.
Very occasionally, after one plant has been collected, the actuator arm positions the plant at the front of the collection box, when the second plant is collected, it collides with the previous plant causing blockage to the turn table. As a result of this and the way the code is implemented (wait until turn table in correct position), the AGV gets stuck during plant collection until the collection box is unjammed.
The solution was sought minutes before the competition, slanted cardboard was added within the collection box, forcing the first plant to slide to the back of the box. The solution was not tested until the competition run. As a result of this change, no blockage was encountered on the competition. A better metal box with built in slanted bottom should have been made to make the solution more elegant.

4 Prototype Production Cost and Scaling

The prodotype production cost is £18,471. With almost the entirety of the cost going towards engineering labour, specifically in the design stage (£10,710). See figure 3 for a complete break down on the cost of the prototype. The cost of the AGV can be significantly reduced through mass production, since the design on the working AGV is a one-off cost. Figure 4 shows the projected unit cost when AGV is in barch/mass production. It is clearly seen when just five AGVs are produced per year, the cost of each individual AGV reduces from £18,471 to £4,500. Admin and service cost increases as the production of AGV increases, however the cost of individual AGV reduces to £1,000 when a sale of 100 units per year is reached, making it very affordable.

5 Appendix

5.1 Costing Summary

IDP Costing Summary

Team	**L102**								
Date	*21/02/18*								
Prototype/Engineering Labour Costs									
	Design		**Construction**		**Testing**		**Admin/Doc**		**Total**
	Hrs	**Cost**	**Hrs**	**Cost**	**Hrs**	**Cost**	**Hrs**	**Cost**	**Hrs (/72)**
Week 1	55.3	£2,810	2	£60	5	£200	16	£320	78.3
Week 2	75	£3,750	10	£300	15	£600	20	£400	120
Week 3	67	£3,350	44	£1,320	17	£680	9	£180	137
Week 4	16	£800	51	£1,530	41	£1,640	12	£240	120
Week 5	0	£0	10	£300	29	£1,160	26	£520	65
Total	**213.3**	**£10,710**	117	£3,510	107	£4,280	83	£1,660	520.3
									(/ 300)
Production Labour costs									
100 off	(each)	£2,943	**50%**	£1,755	**20%**	£856	**20%**	£332	
1000 off	(each)	£1,740	**35%**	£1,229	**10%**	£428	**5%**	£83	
Component Costs									
Mechanical			**Fasteners**	**Sheet Materials**	**Sub-con**	**Pneumatics**	**Gears & Wheels**	**Motors**	**Extras**
Total (One Off)	(each)	**£375**	£5.66	£1.98	£20.00	£160.40	£21	£166.50	£1.44
Total (100 Off)	(each)	**£339**	**90.0%**	**90.0%**	**90.0%**	**90.0%**	**90.0%**	**90.0%**	**90.0%**
Total (1000 Off)	(each)	**£235**	**40.0%**	**50.0%**	**50.0%**	**70.0%**	**40.0%**	**60.0%**	**50.0%**
Electrical			**PCBs**	**Passives**	**Connectors**	**Semis**	**Sensors**	**LEDs**	**Extras**
Total (One Off)	(each)	**£89**	£14.10	£1.48	£15.10	£6.11	£50.00	£1.86	£0.00
Total (100 Off)	(each)	**£80**	**90.0%**	**90.0%**	**90.0%**	**90.0%**	**90.0%**	**90.0%**	**90.0%**
Total (1000 Off)	(each)	**£43**	**50.0%**	**50.0%**	**60.0%**	**75.0%**	**40.0%**	**70.0%**	**50.0%**
Software			**Micro**	**Interface**	**Power Supply**				**Extras**
Total (One Off)	(each)	**£297**	£270	£6	£21				£0.00
Total (100 Off)	(each)	**£240**	**80.0%**	**80.0%**	**90.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
Total (1000 Off)	(each)	**£109**	**35.0%**	**35.0%**	**60.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
Startup Costs									
Mechanical		**£3,000**	*one off*	*[e.g. Tools etc]*					
Electrical		**£2,500**	*one off*	*[e.g. Instruments (Oscilloscope/PS/SigGen...), soldering iron]*					
Software		**£1,500**	*one off*	*[e.g. PC, Software, test boards]*					
Totals									
Prototype		**£18,471**	each		Proto Costs	Component Cost	Post Costs	Net	
				1	£18,471	£761		£19,232	
				2	£9,236	£761		£9,997	
				5	£3,694	£761		£4,455	
				10	£1,847	£761	£4,790	£7,398	
				20	£924	£761	£2,395	£4,080	
Post Development Fixed Costs				50	£369	£761	£958	£2,089	
Staff		**£27,500**	/yr	100	£185	£659	£479	£1,322	
Accommodation		**£6,400**	/yr	200	£92	£659	£240	£991	
Publicity etc		**£7,000**	/yr	500	£37	£659	£96	£791	
				1000	£18	£386	£48	£453	

Figure 3: Costing Summary

5.2 Unit Cost for Mass Production

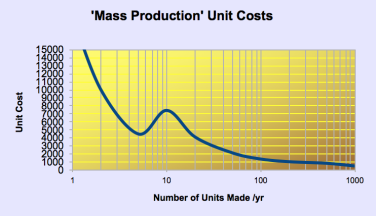


Figure 4: Unit Cost