https://bit.ly/2XmD1Wn ←-- Link to video of the finished product

COLLECTOR

EES 2019

Daniel Vlasits, Martin England, Edmund Goodman, Freddie Ancliff, Ruben Giuliani, and Athena Li

The Perse Upper School Cambridge, in association with Arm Holding

Table of Contents

- Batteries
  - Lead Acid
  - NiMH (Nickel Metal Hydride)
  - Li Ion (Lithium Ion)
  - LiPo (Lithium Polymer)
- Drive systems & Flywheels
  - Brushed DC Motors
  - Stepper Motors (NEMA 17)
  - Windscreen wiper motors
  - Quadcopter Brushless DC Motors
  - Hoverboard Motors
- Input - Process - Output

Budgeting

Hardware Development
- Chassis
- Drive System
- Flywheels
- Ball Ramp
- Cameras and Camera mounts
- Shell Plates

Timeline of CAD Drawings

Health & Safety

Software
- Initial analysis of the problem
- Software layout
- Software development
  - Controlling the output devices
  - Tennis ball analysis
    - Iteration 1)
    - Iteration 2)
    - Iteration 3)
  - Autonomous ball collection logic
- Software testing
- Deliverables
- Version control
- Summary

Introduction

Brief

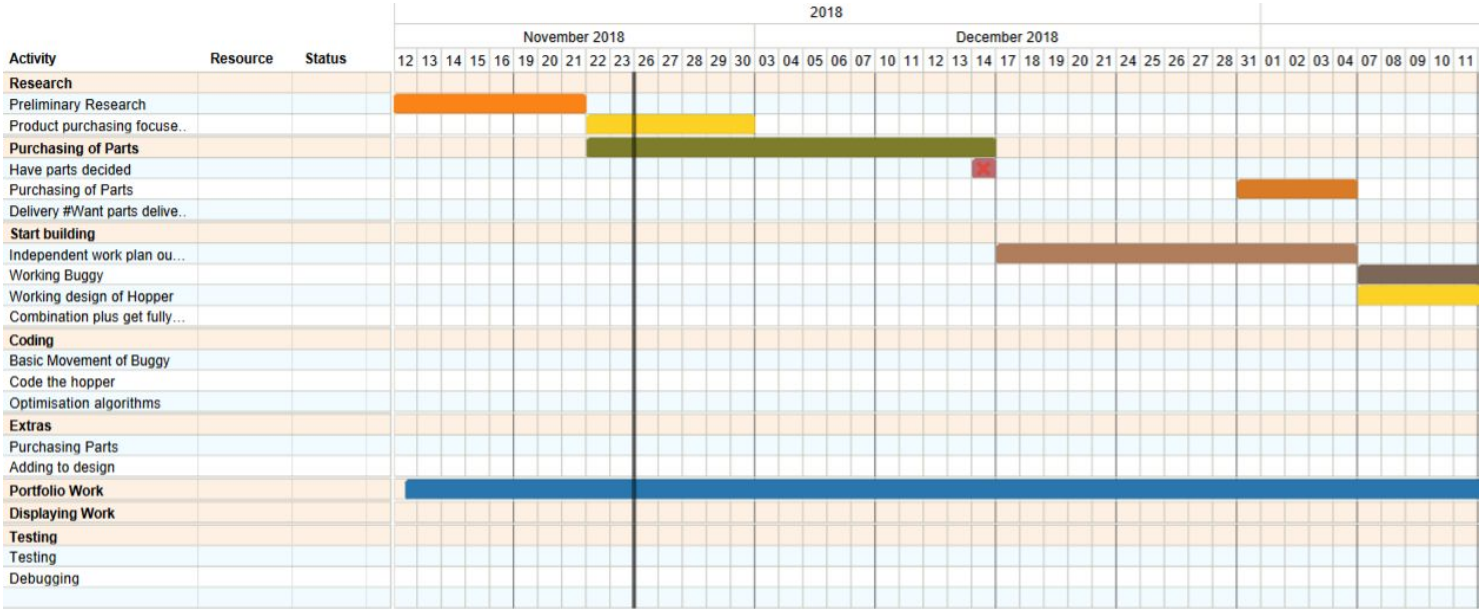
The sports department at our school hold several tennis sessions a day with lots of time wasted at the end of sessions picking up tennis balls, this often leads to a high decrease in the amount of time spent playing tennis. It also means the teachers have to then waste their own time at the end of the day walking around and picking up tennis balls. A tennis ball collector is most useful for education as when a specific skill is being taught hundreds of balls can end up lying around on the other side of the pitch.

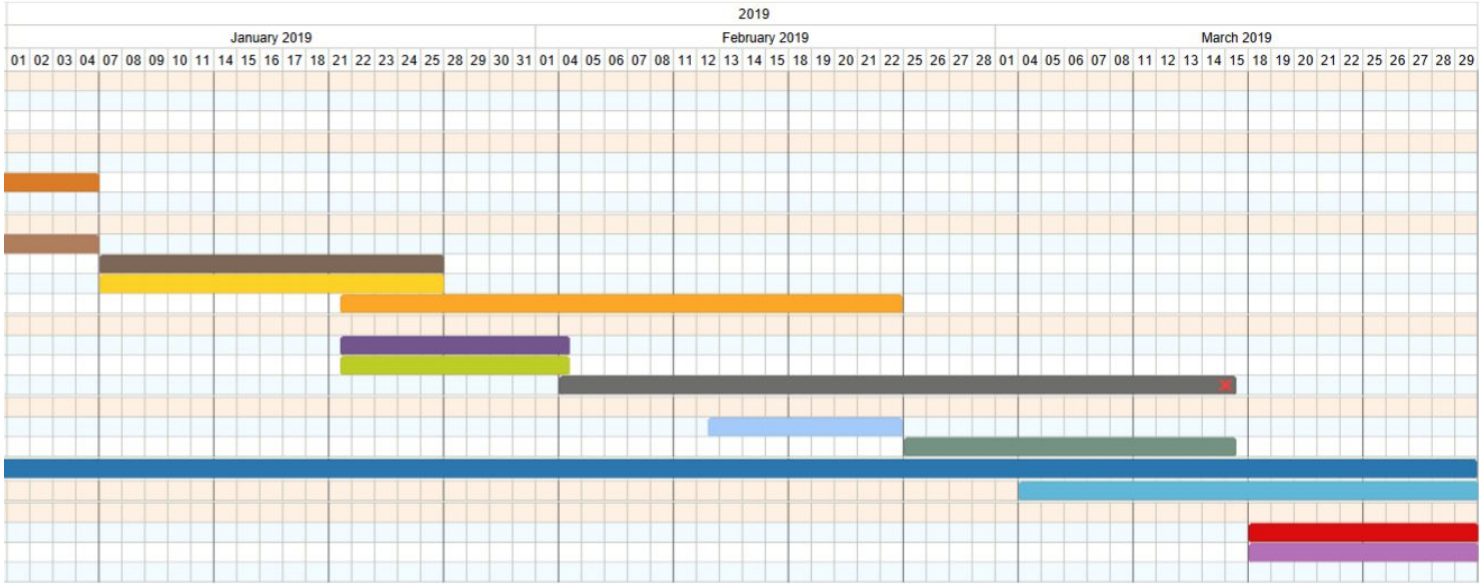
We believe that building a robot to do this autonomously will increase the effectiveness and morale of teachers and students, as they will no longer have to complete the dreary task of picking up countless tennis balls at the end of a strenuous tennis session.

The Perse school has several key values, the two most relevant to new innovation is that health and safety of students is of paramount importance and that we must “value the environment” we have made sure to keep these principles at the forefront of our minds when designing the tennis ball collector.

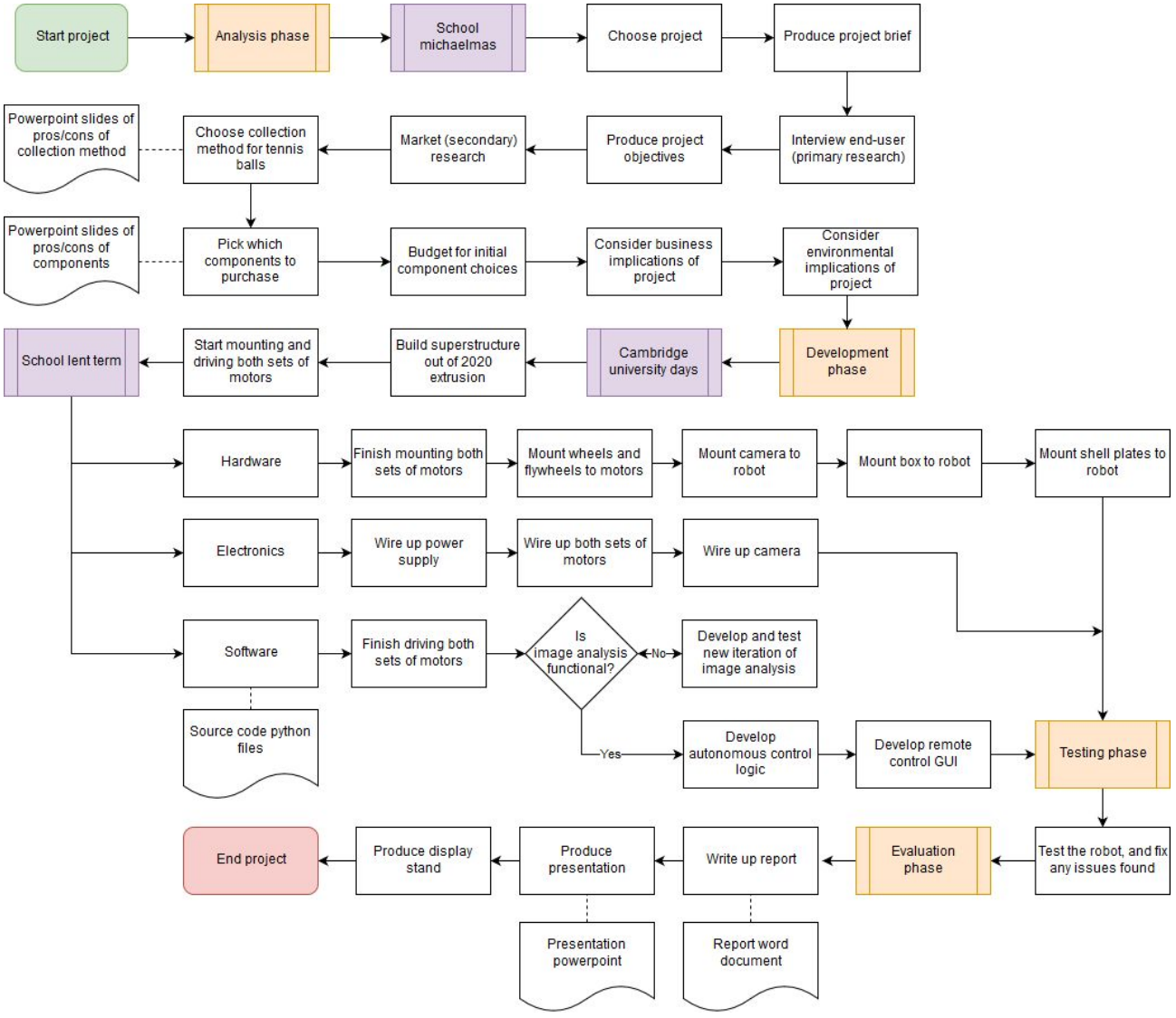
Time management

Gantt chart

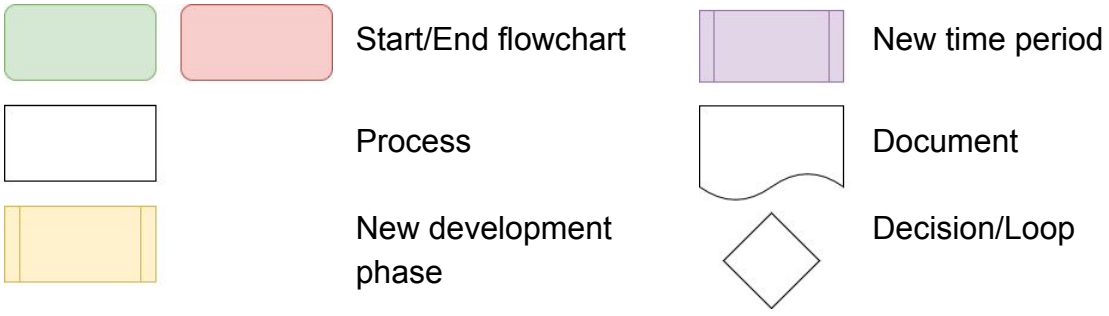




Process map



Key:



Market research

Existing products

There are only a couple existing products on the market at the moment:

Tennis Ball Boy[1]

Design company a Yanko Design[2] produced concept art for an autonomous tennis ball collector. Its promotional text is as follows:

> *"Tennis ball boy is a robot cleaner to pick up a tennis ball, and save people from ball picking up trouble.*
> *The belt made of Velcro material in this product is used to pick up the ball. This product is manufactured drawer type, so you can easily take out the ball.*
> *Moreover, Manual mode is also possible to operate directly." [sic]*

It is not very clear from the promotional images how it functions, but it is stated that there is a removable drawer, so you can easily remove tennis balls, and since a "manual mode" is discussed, it is implied that it is an autonomous machine.

The means of collection is stated to be a strip of velcro, but from experience we note that velcro degrades fairly quickly, and would almost certainly damage the tennis balls during collection. Furthermore, in the concept art it depicts two contradictory methods of ball collection, with the drawer coming out of the side, and an opening lid.

A Vice News article about it[3] suggests that it is useful since it "never tires, and never needs a water break", however, it also asserts that it will not fulfill the job of a ball-boy, as they perform many other tasks, not just collecting tennis balls.

It appears that this is merely concept art, and was never developed as a product, and if it had been, there might be fewer contradictions and physical oversights in its design. However, it does indicate to us that there appears to be a market for a robot of this type.

[1] https://www.yankodesign.com/2014/07/30/the-ball-boy/
[2] https://www.yankodesign.com/about/
[3] https://motherboard.vice.com/en_us/article/8qx8v4/will-tomorrows-ball-boys-and-girls-be-robots

Bear Claw[4]

Students on the University of Berkeley Mechanical Engineering 102[5] produced a ball collecting robot prototype, which they described as:

> *"Tennis courts often become cluttered with balls when they are being used. Our robot can aid in retrieving balls for the tennis athlete training alone or just be released to clean the court of balls which were left behind in play."*

The robot used a camera module for image recognition, an arduino for control, and a pre-built claw arm to pick up the balls, along with two 12V motors to drive it. Furthermore, there is a video of it driving towards, and picking up a tennis ball, so it is clear that it works, and the source code is available online[6].

Since this prototype was designed and built as a group project for a prestigious university second year mechanical engineering class, using mostly out of box components, it is clear that this project is wholly complicated enough, especially since we are not using many out of box mechanical components, and hence is definitely in scope for our EES project.

Other products

In the course of our research, we also found various other products, which we have chosen not to mention in order to improve the succinctness and clarity of our report. Whilst we believe that the two aforementioned products provide enough data to inform our planning, we do also note that other products are available, most of which can be found on Kickstarter.

[4]http://courses.me.berkeley.edu/ME102B/Past_Proj/f09/1%20BearClaw%20Tennis%20Ball%20Collector/index.php
[5] http://courses.me.berkeley.edu/ME102B/index.html
[6]http://courses.me.berkeley.edu/ME102B/Past_Proj/f09/1%20BearClaw%20Tennis%20Ball%20Collector/code.txt

Primary Research

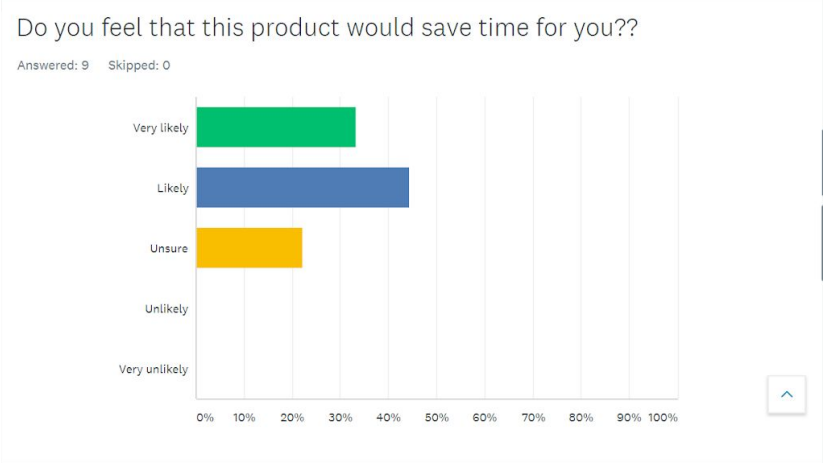
Summary of interview with sports department

Early on in the process, we talked to the sports department to analyse if this would be a viable project, and if it would help them. We produced the following bullet pointed list of their opinions:

- “Really good idea”
- It needs to be able to survive rain
- It would need to be safe and not intimidating “Needs to not kill children”
- There are lots of obstacles left on the pitch that it would need to avoid
- A removable bucket to pour balls into ball basket would be a useful feature
- A house with a button to set it off might be a useful feature

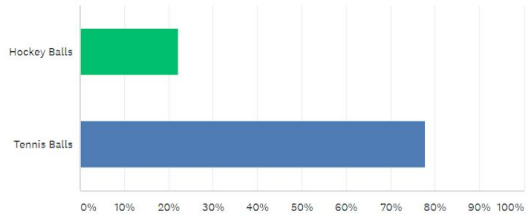
Survey

As part of our market research we conducted a survey among Sports teaching staff to collect their feedback on our intended product. The results are shown below:



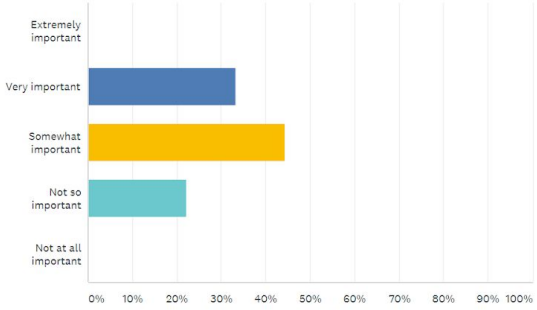
Would this product be more useful to collect Hockey balls or Tennis balls?

Answered: 9 Skipped: 0



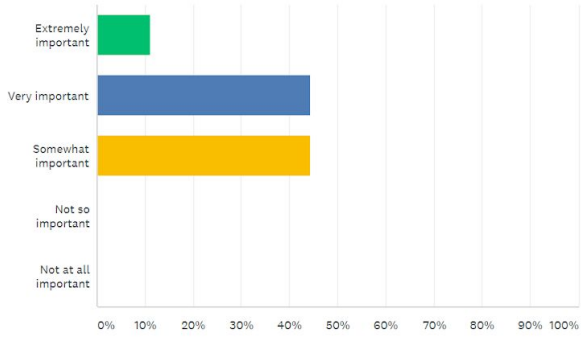
How important is a large hopper capcity?

Answered: 9 Skipped: 0



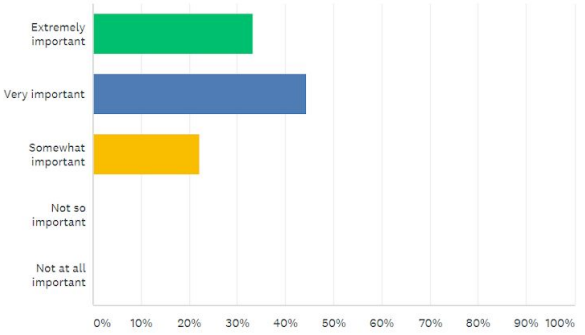
How important is a long battery life?

Answered: 9 Skipped: 0



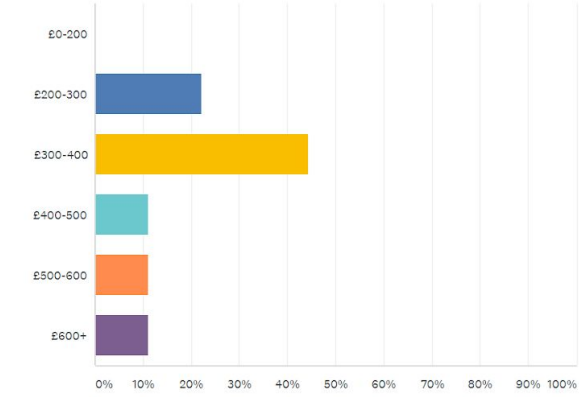
How important are collision avoidance safety systems?

Answered: 9 Skipped: 0



What price would you be willing to purchase such a product for?

Answered: 9 Skipped: 0



As can be seen from the results of the survey, the ball collecting robot would be used primarily to pick up tennis balls. It needs to have storage for quite a few tennis balls however, this is not the most important feature. It should be able to be run autonomously and controlled manually. Furthermore, the average price suggested by respondents was roughly £375. The most important features of the robot should be a long battery life and collision avoidance systems to improve bystander safety during operation.

Business Considerations

Calculation of the cost saved per annum by use in-house

Estimated time spent picking up tennis balls by one sports teacher each session:
5 minutes = 1/12 hours

Estimated number of sessions per day:
2 sessions

Estimated number of sessions per annum, as tennis is only played in 2 terms of the year
12 weeks (1 term) * 2 terms * 5 days * 2 sessions = 240 sessions

Estimated amount of time spent picking up tennis balls by one sports teacher per annum:
240 * 1/12 = 20 hours

Estimated number of sports teachers:
2 teachers

Estimated total amount of time spent picking up tennis balls per annum
20 * 2 = 40 hours

Estimated hourly rate of sports teachers pre-tax
£16 per hour[7]

Estimated cost saved per annum
£16 * 40 = **£640**

Possible revenue by sale to other schools/clubs/companies

It is also possible that the school or ARM could retail the robots to other schools, clubs or companies. This would carry many costs including those of buying components, time spent assembling the robots, and advertisement, so it is likely not profitable. However, following the project, the prototype version might be sold to a tennis club instead, As our product is not professionally designed, it would be sold at a low price. We estimate that a one off sale of **£600.00** might be reasonable, based on the expense of the components, and the time put into development and assembly, however, if we were to produce a batch of robots, we would estimate a lower price, in following with the primary research of **£375.00** - **£425.00**, as the cost of development would be spread across many clients, we would be able to buy components in bulk, and it would be viable to use different manufacturing techniques that are faster, and more cost effective for a batch production process rather than prototyping (see Environmental considerations: Materials and design processes).

[7] https://www.payscale.com/research/UK/Job=Secondary_School_Teacher/Salary

Planning

Project Objectives

After looking at the market research and our survey and discussions with the sports department we compiled a list of objectives that we thought an autonomous tennis ball collecting robot should complete:

1) It should be able to move around without human assistance
2) It should be able to pick up tennis balls without human assistance
3) It should be able to store any tennis balls it picks up, e.g. in a basket or bucket
4) It should be easy to remove/replace the holder for the collected tennis balls, so it can be manipulated individually
5) The holder for the collected tennis balls should be large enough to be of practical use
6) It should be able to autonomously detect tennis balls, and drive towards them
7) It should be able to autonomously detect walls and other solid obstacles and avoid them
8) It should be water / weather resistant
9) It should be able to function autonomously in different weather conditions, e.g. the camera should function in low and high brightness conditions
10) It should be able to drive at a reasonably high speed, e.g. a walking pace, so it can collect balls in a fairly short period of time
11) It should have an intuitive user interface, either physically on the robot, as a software package, or both
12) There should be a manual mode to allow operators to control the robot directly
13) It should be able to collect most of the balls left on the pitch, and not leave a large number lying uncollected
14) It should be rechargeable
15) It should be easy to maintain/repair
16) It should have a fairly low profile, and be fairly aesthetically pleasing
17) It should not be intimidating, i.e. not making very loud noises, not travelling excessively fast, and its autonomous algorithm should treat people as obstacles, so avoid them
18) No Exposed Electronics

ACCESS FM analysis

Specification area	Requirement	Consideration
Aesthetics	The robot must not be intimidating	The robot should not move excessively fast
Customer	The robot must be easy to control	The robot should have an intuitive user interface
Cost	The robot must be within budget	Buy components of an appropriate price
Environment	The robot must be weatherproof, and able to function in various weather conditions	Produce shell plates to cover electronics
Size	The robot must be large enough to carry a large number of tennis balls	The robot should carry a large box for tennis balls
Safety	The robot must be safe to operate in a public area	The robot should have no exposed wires, and avoid people
Function	The robot must be able to autonomously collect tennis balls	Implement and test software to autonomously collect tennis balls
Material	The robot must be weatherproof, and strong enough to hold a large number of tennis balls	Use a weatherproof, durable material, or apply a weatherproof coat to the material

Ball Collection Methods

We considered various possible designs for our robot, and used the acronym ACCESS FM (Aesthetics, Customer, Cost, Environment, Size, Safety, Function, Material) to assess each design, by giving each attribute a score out of ten, then selecting the design with the highest score.

Flywheels

Specification Table			
Point	**Score**	**Point**	**Score**
Aesthetics	6	Size	7
Customer	7	Safety	6
Cost	8	Function	8
Environment	6	Material	7
		Total Score	55/80

Two flywheels with a rubbery surface rotating at high speeds and tilted at 45 degree angle would grab onto balls funnelled towards them and shoot them behind into the collection crate.

- Pros
  - Will only collect balls, automatically ignoring leaves or other debris on court as custom width between flywheels can be set.
  - Easy to store the collected balls. The velocity generated by the flywheels can be used to send the balls in any direction. For example flick them up into a basket, or simply shoot them into a bag dragging behind without the need of a separate acceleration system.
  - Relatively easy to put together and/or integrate into a pre-existing buggy.
  - Can quickly collect many balls with low risk of jamming, especially combined with a funnel-shaped ball guide.
  - Very easy to program, as it is just driving motors
- Cons
  - Needs a lot of battery power as flywheels need to keep rotating even when not actively collecting balls, low efficiency.
  - It could be loud, especially if the flywheels are unbalanced
  - Cannot deal with balls stuck in corners or even near edge of court, especially if using funneling mechanism - there could be a way however to have the flywheels on an extended section at the front which could reach into corners and grab balls at a distance.
  - Definitely requires extra motors on top of the ones for the drive.

Rotating Collector

Specification Table			
Point	**Score**	**Point**	**Score**
Aesthetics	6	Size	6
Customer	7	Safety	7
Cost	7	Function	6
Environment	8	Material	6
		Total Score	53/80

The two front wheels are connected by an axle with several ridges around which there is a chamber to restrict the space (imagine a revolving door). The ball gets carried by the rotating axle and is forced out at the back into a container.

- Pros
  - Very efficient, ball just gets carried with the motion of the axle and doesn't need an acceleration system to get it into the container, saving a lot of energy.
  - Very easy to program, as it is just driving motors
  - In an ideal situation with many balls in front of it it would easily pick up many compared to other systems. This would never be seen if collecting one ball at a time but if combined with the idea of pushing all the balls to one area and then collecting them it could be useful.
  - Much quieter than flywheels
- Cons
  - The system would require very high torque motors, as they would need to both drive the robot forward, and spin a relatively massive drum, used to collect the balls
  - It may be difficult to design and build a drum that is durable enough to deal with jams and debris without breaking
  - Would not be able to collect balls from corners or near edges as ball needs to be practically under the wheel before it starts being lifted.

Robotic arm

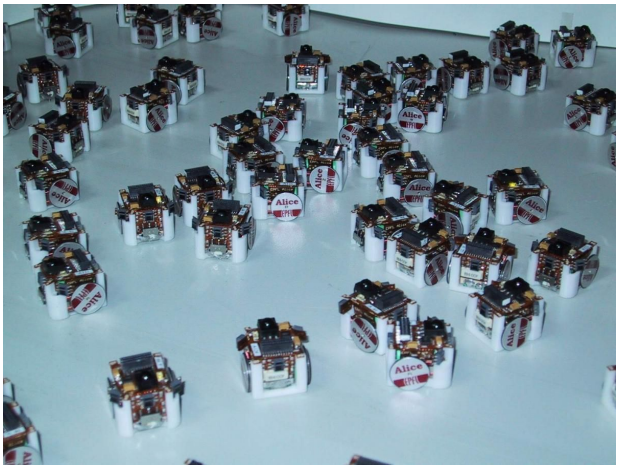
Specification Table			
Point	**Score**	**Point**	**Score**
Aesthetics	8	Size	5
Customer	8	Safety	6
Cost	5	Function	4
Environment	7	Material	6
		Total Score	49/80

A fully automated robotic arm would be attached to the buggy and pick up balls individually.

- Pros
  - Would be able to pick up balls in corners and near edges which the other methods would struggle with. The only one that can do this and probably worth it just for that.
  - Very quiet.
  - Uses relatively little energy compared to the other systems in which a motor needs to keep rotating throughout the whole ride. A robotic arm would only require some precise movement every once in a while.
  - Could be expanded to have multiple robotic arms on the buggy which would speed up the pickup time in areas dense with balls.
- Cons
  - If it is going to work at all it needs to be very well made.
  - Really hard to make, both mechanically and with the software.
  - Would require multiple cameras (or 360 camera), digital gyroscope(s), high accuracy infrared range finders, gps etc.
  - The processing power needed to compute all these inputs would be quite substantial
  - Collection would be quite slow compared to the other systems in areas with many balls

Swarm of mini robots

Specification Table			
Point	**Score**	**Point**	**Score**
Aesthetics	6	Size	5
Customer	5	Safety	5
Cost	2	Function	7
Environment	5	Material	6
		Total Score	41/80



We would have mothership with the ball container which stays still and deploys an army of small fast moving robots each in charge of collecting one ball from the court which they would then return to the base.

- **Pros**
  - Probably the quickest pickup time out of all, the small size and the maximum load would allow for the mini robots to travel at high speeds and cover a large area in a small amount of time.
  - Would be able to collect balls pretty much anywhere in the court, obstacles are not a problem and neither are corners
- **Cons**
  - All the problems with the robotic arm, times however many mini robots we want.
  - It would require a lot of effort to produce even a small set of robots
  - Pretty loud to have many small brushed dc motors running at the same time.
  - Charging is an issue, there would have to be an automated charging bay in the mothership.
  - Stray robots getting lost.
  - Safety concerns.

Sweeper Arm

Specification Table			
Point	**Score**	**Point**	**Score**
Aesthetics	6	Size	6
Customer	5	Safety	6
Cost	6	Function	6
Environment	6	Material	7
		Total Score	48/80

Some kind of arm coming out from the side would be used to push all the balls to a certain area after which we either leave it to a human to do the rest or integrate a collection mechanism to collect all the balls itself. This sort of idea could also be combined with a funneling mechanism and the flywheels/rotating axle to allow the latter mechanisms to collect balls from corners and edges.

- **Pros**
  - If used in addition to the flywheels it could result in a perfect solution to the corner/edge problem,while maintaining the simplicity of the mechanism.
- **Cons**
  - Probably more complicated than it seems to send a ball into a precise area from a distance.
  - Tennis courts are very big, not so easy for such a small machine to apply such a great force, especially an angular one.
  - The software is not simple either, to find individual balls and position the buggy at the right distance and angle to propel it in the right direction and at right speed is almost more complicated than just picking it up.
  - A robot blindly launching tennis balls across a field is a threat.
  - Even when used to dig balls out of corners it is not an easy task and if not made well there is a risk of it getting stuck, breaking or not working.

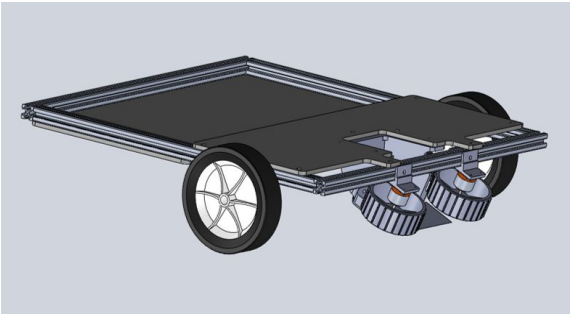
Final design

We chose to use flywheels as they scored highest in our ACCESS FM analysis, and we thought that they were the overall best choice in terms of both effectiveness and ease of implementation. There were also many other possible designs we could have used, but did not fully consider. For example, we initially thought a Velcro based system might be effective, but then noted it would likely damage the balls, meaning it was neither cost effective, nor environmentally viable. Various other solutions seemed initially effective, but we quickly veteod due to various flaws.

Development

Hardware

In the end we chose flywheels paired with a funneling system as our collection mechanism.

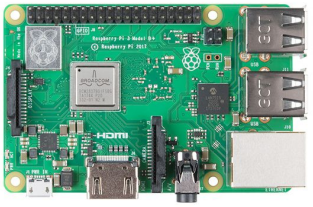


Justification:
Scored the best in the spec table assessment and seemed to have the most advantages out of all the possible systems. Pragmatic thinking was key in reaching a conclusion as we had to be careful to choose a system which was evidently feasible to manufacture with our available resources, even if it meant to forgo optimal functionality.

Control Board

For the control board, we had to select a microprocessor which was inexpensive enough to fit in our budget, computationally powerful enough to control our robot, including performing image analysis, but also had enough GPIO pins to support our input and output, and a way to connect a camera. We considered various options, including the Raspberry Pi 3B+, Arduino Uno, and Mbed NXP, and ranked them based on their fulfillment of our criteria.

Raspberry Pi 3B+



Specification:

- 1.4GHz CPU with 1GB SRAM, and a VideoCore IV multimedia/3D graphics core @ 400MHz
- 28 GPIO pins, a camera interface, 4 USB type A ports, HDMI output port
- Fully fledged OS[8], which allows running of python, and easy downloading of libraries
- Wifi enabled, allowing SSH and VNC remote connections

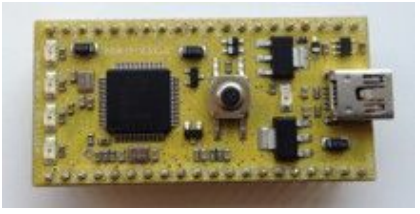
Advantages:

- Comparatively high processing power, meaning more complicated image analysis is viable
- Easy to connect to via wifi, and can be used as a computer with a monitor and keyboard connected

Disadvantages:

- Comparatively expensive

Mbed NXP



Specification:

- 66MHz CPU with 32KB flash, 8KB SRAM
- 9 GPIO pins, and a USB type mini-A connector
- Runs C/C++ variant

Advantages:

- Fairly inexpensive, an ARM holdings product, so linked to our associate company

Disadvantages:

- Fairly low processing power, meaning more complicated image analysis is not at all viable
- Cannot be connected to via wifi, only via USB mini-A connector
- Can only be programmed in C/C++ variant, which has fewer libraries, and we are not experienced using

Arduino Uno

Specification:

- 16MHz CPU with 32KB flash, 2KB SRAM
- 20 GPIO pins, and a USB type B connector
- Runs C/C++ variant

Advantages:

[8] https://www.raspberrypi.org/downloads/raspbian/

- Durable, and commonly used by hobbyists

Disadvantages:

- Fairly low processing power, meaning more complicated image analysis is not at all viable
- Cannot be connected to via wifi, only via USB type B connector
- Can only be programmed in C/C++ variant, which has fewer libraries, and we are not experienced using

We eventually settled on the Raspberry Pi, as we decided it best fulfilled our criteria. In comparison with the Mbed and the Arduino, it was clearly a stronger choice, as it has a much faster clock speed (1.4GHz >> 66MHz and 16MHz respectively), much more RAM (1GB >> 8kB and 2kB respectively), many more GPIO pins (28 > 9 and 20 respectively) and a camera port, all of which allow it to excel at image analysis, a key part of our control flow. Furthermore, it supports python, the language we are most familiar using, and has a fully fledged OS, which will speed up the development process.

Batteries

For the batteries, we had to select a battery which was inexpensive enough to fit in our budget, able to provide an appropriate amount of current and voltage, of a large enough capacity to be able to power our high current draw robot for some time, and environmentally friendly and safe enough for us to use.

Lead Acid

Advantages:

- Available in a wide variety of shapes and sizes
- Good value for a large amount of power
- Withstands slow, fast and overcharging without dangerous effects
- Long life cycle
- Very low internal resistance, so high peak discharge current, and minimal loss of energy due to internal resistance

Disadvantages:

- Very heavy due to the lead which is quite dense
- Low weight to energy ratio
- Can’t output large amounts of current for extended periods
- Must be stored in charged state to prevent damage (sulfation)

NiMH (Nickel Metal Hydride)

Advantages:

- Relatively high power density and capacity (compared to lead acid)
- Safer chemistry than lithium
- Less prone to memory effects
- Simple storage/transportation
- Environmentally friendly

Disadvantages:

- Can't output as high currents as Lithium batteries - larger currents reduce lifespan
- Quite expensive compared to other batteries for the same capacity
- Requires slightly more complex charging than Lead Acid

Li Ion (Lithium Ion)

Advantages:

- Highest energy density
- Low self discharge over time
- Low maintenance as they don't need to be periodically recharged
- Safer than LiPo

Disadvantages:

- Can't output as much current as LiPo
- Require special equipment to attach cells together (spot welder)
- Require cell protection and balance charging to prevent damage
- Limited number of charge-discharge cycles

LiPo (Lithium Polymer)

Advantages:

- Very high current outputs due to low internal resistance (>200A for some batteries)
- Available in a wide variety of shapes, sizes, voltages and capacities
- High power density
- Less expensive than NiMH or Li Ion of the same size

Disadvantages:

- Soft pouch design can be easily punctured by sharp implements resulting in fire
- Overcharge/discharge can lead to cell failure and/or fire

- Require a balancing charger to ensure and even voltage in all cells
- No protection circuits built in to large batteries due to the high current output

We eventually settled on the Lead Acid battery, as we decided it best fulfilled our criteria. The lead acid battery can output very high peak currents, and is quite efficient, due to its low internal resistance, however, it is much safer than LiPos, which are the only other battery able to yield a similar power. We thought that its drawbacks, such as its weight, did not affect our use case, and all of the other types had drawbacks that would be more disadvantageous in our project. We noted that it might seem environmentally harmful, but found that if disposed of properly, it is more environmentally friendly that other batteries, due to its high efficiency, from its low internal resistance (see environmental considerations)

Drive systems & Flywheels

For the drive systems and flywheels, we had to select motors which were inexpensive enough to fit in our budget, and very high torque and high speed respectively, as otherwise our robot would not be able to move, nor pick up tennis balls.

Brushed DC Motors

Specification:

- Output Speed: 1000 RPM
- Voltage: 12V
- Dimensions: 106x37 mm

Advantages:

- Require only inexpensive motor drivers to run both forwards and backwards.
- Don’t require very much current (less powerful battery needed)
- Doesn’t require any additional gearing due to built in gearbox

Disadvantages:

- High speed geared DC motors are fairly expensive (>£10 each)
- They are quite loud in operation due to metal gears and motor brushes
- Can’t operate efficiently and a large range of speeds

Stepper Motors (NEMA 17)

Specification:

- Max Output Speed: ~900 RPM (Difficult to find reliable data)
- Voltage: 5V
- Dimensions: 42x42x40 mm (case) 4mm shaft diameter

Advantages:

- Very high precision in terms of position

- Good low speed torque

Disadvantages:

- Torque drops with the square of the speed (motor can easily slip when at high speed)
- Low efficiency and so can get hot
- Very noisy at high speeds

Windscreen wiper motors

Specification:

- Max Output Speed: ~50 RPM (Difficult to find reliable data)
- Voltage: 12V
- Dimensions: 198x108x98 mm

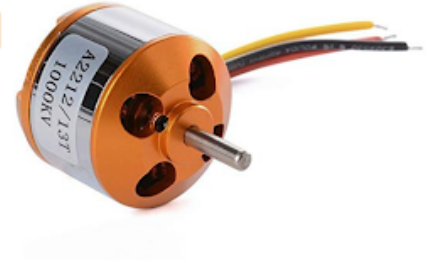
Advantages:

- Very high torque
- High gear ratio, and non-backdrivable, due to internal worm gear

Disadvantages:

- Draw a high current, so difficult to find appropriate motor drivers

Quadcopter Brushless DC Motors



Specification:

- Output Speed: 1000KV (12000 RPM at 12V)
- Voltage: 7-12V
- Max. Current Draw: 12A
- Dimensions: 27.5x27mm (case) 3.2mm shaft diameter

Advantages:

- High torque at wide range of speeds
- High efficiency (80%)
- Quieter than many other motors

Disadvantages:

- Require speed reduction for driving wheels (e.g. belt and pulleys)
- Motor drivers that are bidirectional are fairly expensive (>£9)
- Require a battery that can deliver a large burst current

Hoverboard Motors

Specifications:

- Max Speed (with 6.5” wheels): 15 km/h or ~480 RPM
- Voltage: 36-48V
- Dimensions: 165mm diameter

Advantages:

- Wheel is pre-attached so no gearing/extra components needed other than mount
- Very powerful and fast (can carry a person)
- Can be controlled with ordinary brushless ESCs

- Quite compact as the motor is within the wheel

Disadvantages:

- Expensive (~£20 each plus £9 driver)
- Lack of internet data due to being part of a commercial product
- Require a large, high voltage, high power battery (Li-ion)

We eventually chose windscreen wiper motors to drive our wheels, and quadcopter motors for our flywheels, as we thought they best satisfied our requirements of being high torque and speed respectively, they were fairly inexpensive, and are very commonly used/tested, so we were fairly confident they would not break.

Input - Process - Output

Inputs	Process	Outputs
Webcam: Takes a photo of the pitch in front of the robot, so we can locate and drive towards tennis balls 	Raspberry Pi: Reads in data from the webcam and time of flight sensors, in order to ascertain which direction to drive the robot, and when to spin up the flywheels.	Windscreen wiper motors: Drive the wheels, moving the robot 
Time of flight sensor: Gives a distance to the nearest wall, allowing us to avoid crashing into them 		Brushless motors: Drive the flywheels, allowing us to pick up the tennis balls 

Budgeting

Item	Quantity	Cost for 1 item	Total cost
Raspberry Pi 3B+	1	£32.38	£32.38
12V 22AH Lead acid battery	1	£43.99	£43.99
Windscreen wiper motors	2	£17.00	£34.00
4x 75mm Heavy duty rubber swivel castor wheels	1	£6.39	£6.39
RPi ,motor driver board MC33886	1	£23.20	£23.20
2020 Aluminium extrusion 500mm length	4	£6.69	£26.76
Pi heatsink 3pcs	1	£3.39	£3.39
Sandisk micro SD card + adapter	1	£5.99	£5.99
Pi camera	1	£10.77	£10.77
Pi camera extender	1	£1.28	£1.28
VL53L1X Time of Flight (ToF) Sensor	1	£9.46	£9.46
1000kV Outrunner brushless motors + 30A ESC	2	£13.64	£27.28
Rubber wheel 150mm diameter	2	£2.99	£5.98
T-nuts for aluminium extrusion	1	£8.29	£8.29

This came out to a total cost of **£239.16.**

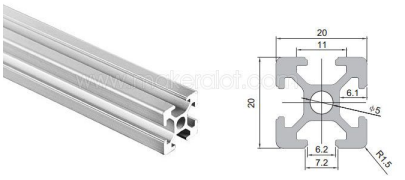
Over the course of the 6 months we were advancing rapidly in the design and build phase that to develop the best, most useful robot for the the school we needed to invest in a few additional parts. These parts were bought by the school as the robot was showing such promise:

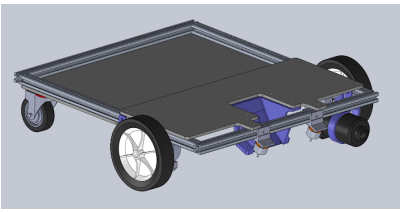
Item	Quantity	Cost for 1 item	Total cost
Diablo Motor Controller	1	£66.66	£66.66
PIXY 2 camera	1	£55.99	£55.99

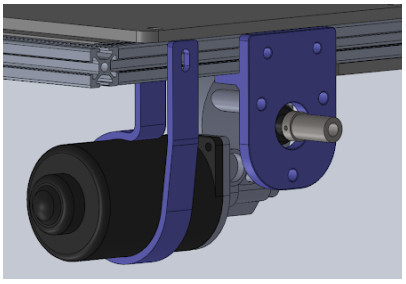
This brought the total cost for development of the robot to **£361.81.** Many parts were used along the way to prototype the design which were available to us within the school including relay boards**,** 3d printing filament and mdf. Links to the exact parts bought can be found in the appendix 1: Links to budgeted components.

Hardware Development

Chassis



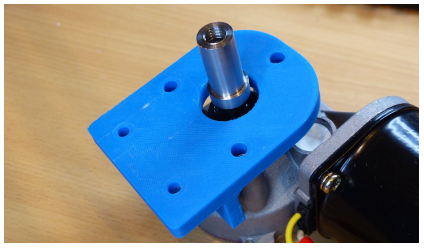


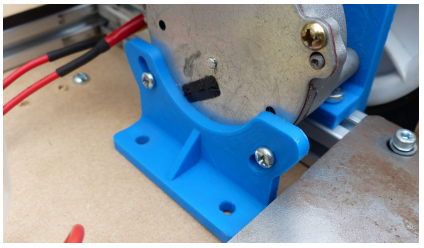


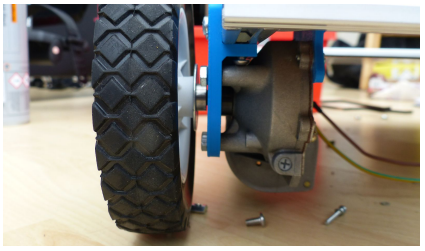
To provide rigidity and allow for modularity in arranging components, we built the main frame from aluminium extrusion. We used 4x 500mm lengths of 2020 extrusion attached to form a rectangle using right angled brackets and T nuts which fit into the rails. The metal and MDF sheets sandwiched the extrusion on the top and bottom to mount electronics and the basket for tennis balls.

The motors were mounted with 3D printed brackets attached to the extrusion. At first, we only had one holding the very front of the motor however we decided to add another at the back as well as a clamp along the main motor body as the weight of the robot caused the bracket to flex which caused issues while driving.

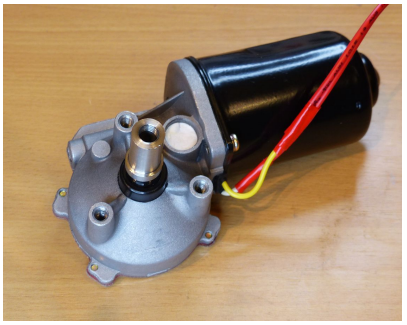
The flywheels were mounted on the front piece of extrusion which allowed us to adjust the distance between them easily so that they would be able to fire the tennis ball up the ramp.







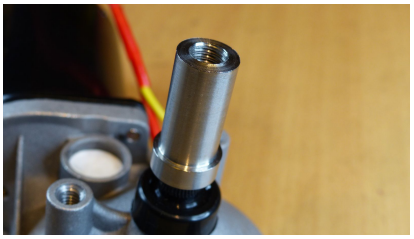
Drive System

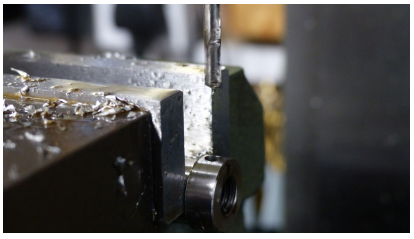


We chose to use windscreen wiper motors to drive the robot as they have a built in gearbox and are fairly easy to control. Although they aren't as fast as some other motors, the worm drive reduction means they have plenty of torque which was important as our robot would end up being rather heavy.

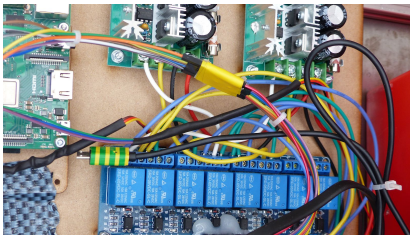
The wheels were 150mm in diameter and were simply bought from screwfix as they were fairly inexpensive at £3.00 each and were very

durable. Since the wheels weren't designed to fit on this motor, we had to make an adapter so they would fit securely. A lathe was used to machine some stainless steel parts with an M8 thread on either end. It screws onto the motor shaft and can be tightened onto the tapered section of the shaft preventing it unscrewing easily. The wheel then slides over the adapter and some M8 threaded rod is screwed into the other end of the adapter. A nut and washer then simply clamps the wheel onto the adapter. Later in the design, we added a grub screw to prevent the adapter unscrewing from the motor shaft.

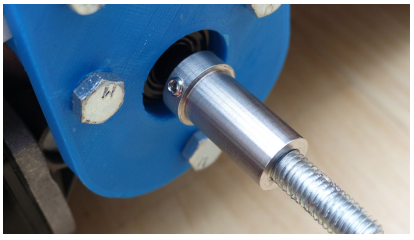




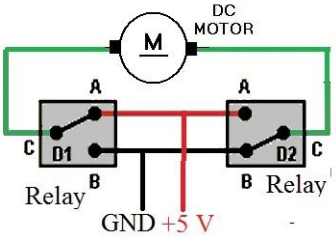
To control the motor, we wanted to use a commercially available motor driver board, however the current drawn by the motors was too high at around 7A while most motor drivers can only handle 5A maximum. Instead, we used a relay board to control the motors as we thought it would be more reliable than trying to build our own MOSFET control board.

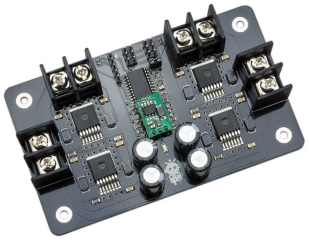


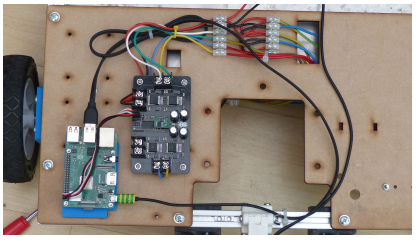
Since this prevented us using PWM to control the motor speed, we added two voltage converters to allow us to precisely adjust the speed of each motor so they would be the same and the robot would drive in a straight line. We used a 3 relays per motor to form a H bridge and to turn each motor on and off, allowing us to run them in opposite directions when it needed to turn on the spot.



Although we used this relay solution for a while during testing, it seemed a bit clunky and over complicated meaning wires could become disconnected causing the robot to stall. Therefore, we decided to invest in a more expensive motor driver board that could handle the current necessary for these motors. This is probably what we should have done from the beginning but that is just part of the development process.

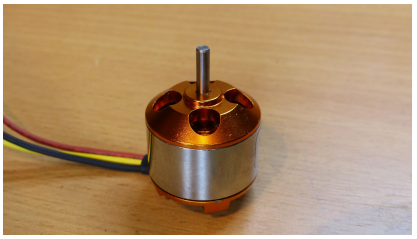






Flywheels

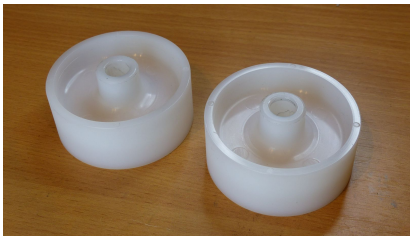
The flywheels are powered by two outrunner brushless motors which are usually intended for quadcopters. We had to make a set of flywheels that could be attached onto these motors that could withstand the forces of squeezing a tennis ball while running at high speeds. The motors come with their own driver called an ESC (Electronic Speed Controller) which can be controlled with a PWM signal similar to a servo.



For the first attempt we quickly 3D printed some simple flywheels in PLA to see if they would work. However, as soon as we first spun them up we realised they were very imbalanced as there was a lot of vibration and it seemed as if they would shatter if we fed a ball into them.

This was likely due to warping of the part meaning it wasn't quite circular as well as the imprecision of the mounting method used. Overall, we decided to abandon these wheels and try a different approach.

A key issue with the first version was the imbalance and lack of strength of the flywheels so rather than 3D printing them, we decided to buy some nylon casters and machine them to give a flat surface. We decided not to add any extra features to improve grip to prevent it blowing air or making too much noise. We figured that allowing the tennis ball to be squeezed by the flywheels would provide sufficient grip.



To the motor adapters, we machined some aluminium parts that clamped the nylon in a similar way to the drive wheels although care was taken to ensure as tight a fit as possible so that the centre of rotation would be correct and there would be minimal wobble as the flywheels span.

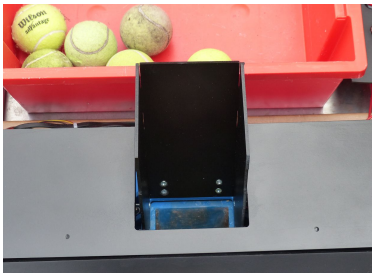
Unfortunately, these wheels still didn't run perfectly balanced and so make a lot of noise when they are running. To partly solve this issue, we pushed the wheels closer together which causes the tennis ball to squash more as it passes through them, therefore providing more grip. This allowed us to run the motors at a lower speed and still successfully pick up tennis balls. We also limited the time the flywheels were on for (which also saves power) and made sure all the components were solidly mounted so they wouldn't rattle which is what made a lot of the noise.

Ball Ramp

The ball ramp consists of a lower scoop which directs the ball upwards and an upper tube that directs the ball into the container.

We originally just 3D printed the scoop from PLA however we soon realised that this would not be strong enough to deflect a high speed tennis ball.

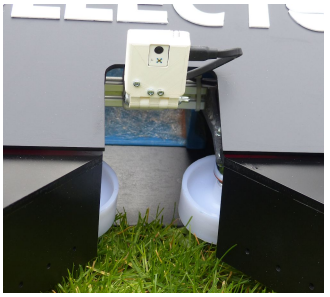
For the second iteration, we added lots of reinforcing ribs to prevent layer delamination. We also printed it in ABS and reinforced key areas with fibreglass to provide a hard surface and reduce the chance of layer delamination even further. Attached to the scoop is an MDF plate which extends down close to the ground to guide the ball into the scoop.



The upper tube consists of another 3D printed bracket with 4 MDF plates screwed to it. The plates also had finger joints and were glued together with wood glue to provide further support at the top. The upper tube prevents the ball from landing on the electronics and guides it into the box. The tube is at a fairly steep angle so that the ball doesn't fly over the box as it exits the tube.

Cameras and Camera mounts

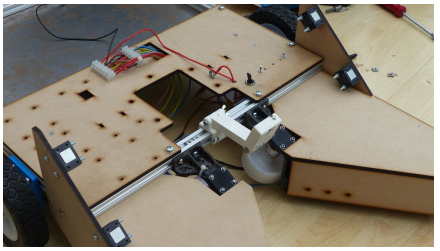
We used a camera to detect where the tennis balls were. We originally used a standard raspberry Pi camera but later switched to a camera called the Pixy as our first camera experienced a hardware failure.



To mount the camera, we used a 3D printed arm made of 3 pieces that were locked together with bolts and wingnuts. This allowed the camera to be repositioned and tightened by hand allowing us to quickly move it around to get the best possible field of view. The mount was simply attached to the aluminium extrusion using a few screws and T nuts.

Shell Plates

The plates cover the electronics, helping to weatherproof the design and make it look more attractive. They are made from 6mm MDF sheet using a laser cutter making them very sturdy.

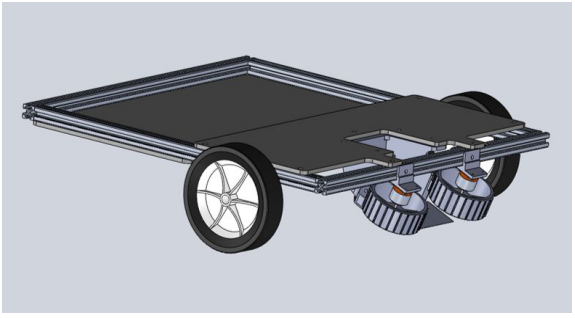


The plates were designed on solidworks so they could be fitted into the main assembly before cutting to ensure they were exactly the right shape and wouldn't interfere with any other components. After cutting, the plates were attached with wood glue and some 3D printed brackets. We used magnets to hold the top and front plates on so they could be removed quickly during testing to access all the electronics.

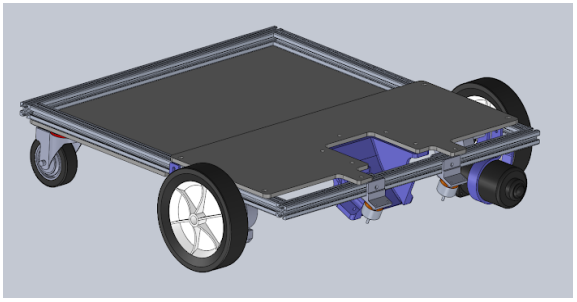
Before painting, P38 Body filler was applied to the surface and then sanded to fill in the rough surface of the MDF. High build primer was then added and lightly sanded after drying with 1200 grit sandpaper. This gave us a much smoother and harder surface which should also be more waterproof than the standard MDF. We then added several coats of matte black spray paint and used frog tape to paint some white stripes. The logo was 3D printed and painted white to match the stripes. These were glued onto the front with epoxy before lacquering all the plates to protect the paint from scratches.

Timeline of CAD

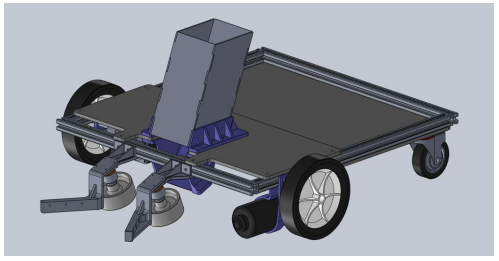
We made a solidworks assembly of the whole robot (except some electronics) so parts could be designed and test fitted before printing them which saved time and materials especially when making the shell plates for the robot. Over time, the assembly got more and more detailed as we added more and more parts to the robot.



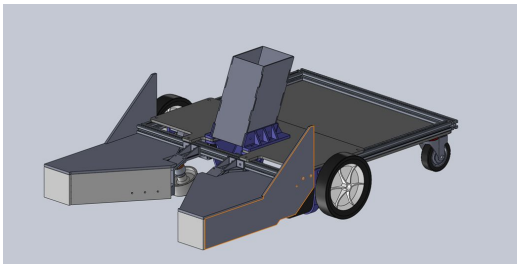
November 2018: Preliminary CAD, modelling the main frame, wheels, and flywheel mounts and motors, along with the initial ribbed flywheels, before the university of cambridge development days



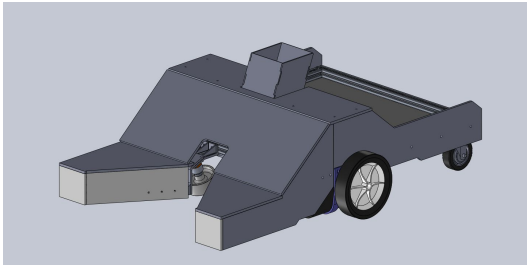
January 2019: The next iteration of the CAD model, with the casters, drive motors and mounts, the ball ramp strengthened, and the ineffective initial flywheels removed, after the university of cambridge development days



February 2019: The next iteration of the CAD model, with the secondary ball ramp, updated flywheel mounts and flywheels, and mounting points for the funnel, during the lent term



March 2019: The next iteration of the CAD model, with the finished ball funnel



April 2019: The next iteration of the CAD model, with all of the shell plates added, covering the electronics

April 2019: The final rendering of the CAD model, with the logo added

Health & Safety

Situation	Hazard	Precaution
3D Printer	Burns caused by touching hot bed/extruder Fumes from printer damaging health	Keep hands away from printer while it is warm Open a window while printing to evacuate fumes
Laser Cutter	Fumes from cutter damaging health	Turn on extractor while cutting parts
Drilling	Swarf from drill getting caught in eyes Loose clothing catching in spinning drill	Wear safety goggles while drilling parts Ensure loose clothing is secure and don't wear gloves
Soldering	Inhaling solder fumes damaging health	Try not to breathe in fumes and open a window
Fiberglass	Resin getting on hands causing skin irritation Inhalation of fibers, resin fumes or dust while sanding causing respiratory problems	Wear gloves while handling resin and fibers Wear a respirator while processing fiberglass
Lathe/Milling Machine	Swarf from metal getting caught in eyes Clothing or hands getting caught in spinning sections of the machine	Wear safety goggles while turning/milling Keep hands and loose cloth away from spinning parts and use included protective shields

Hardware Testing

First tests began at the Cambridge University for the two days when we were able to work there. We tested the first version of the flywheels where we discovered that they were far too fragile and unbalanced. We also tried driving the robot around however since we were using a cheap substitute motor driver, we weren't able to get it driving well and one wheel seemed to spin faster than the other.

To solve this, our first plan was to use relays and voltage converters (see Hardware Motor Control Section). Daniel spent some time wiring this up until he got it working. The voltage converters allowed us to trim the speed of each wheel until they span at the same speed which helped the robot move in straight lines. However, we then discovered that the weight of the battery, chassis and other components caused the drive motor mounts to bend outwards which caused further issues while driving. After a number of attempts, we finally fixed this with a second motor mount at the back of the motor (see Hardware Chassis Section).

When we first tested the flywheels, they made a lot of noise which was because a part which mates with the tapered collet on the motor shaft wasn't pushed in straight. To try to solve this, we machined the aluminium parts down slightly while holding them by the shaft mount. This helped, although the flywheels still weren't super well balanced and wobbled a bit. If we were to do this again, we should spend a bit more time making these perfect however the flywheels function correctly so the noise and vibration is the only issue we have encountered with them.

Following the hardware fixes, Edmund and Daniel spent many weeks working on code that worked increasingly reliably at both detecting tennis balls and driving towards them to pick them up (See Software Development Section). Eventually they got it working very reliably and, although occasionally it didn't pick up a ball correctly, this was not an issue as the random collection strategy would mean that it should eventually try to pick it up again.

Overall, we have had a lot of failures and issues during testing and have replaced most of the parts at least once with upgraded versions which demonstrates the high level of development and improvement of our product.

Software

Initial analysis of the problem

Our project brief is fundamentally to produce a robot that autonomously collects tennis balls. In order to achieve this, we needed to be able to both control the physical hardware of the robot, and evaluate higher level code to control where the robot should move, and analyse images to ascertain if they depict a tennis ball.

We decomposed[9] the task into smaller chunks which were each more easily solvable, and we could write separately. We identified the following subtasks:

- Driving the motors to move the robot
- Driving & calibrating the motors to spin the flywheels
- Detecting and avoiding walls using the time of flight sensor
- Detecting and picking up tennis balls using the camera

Before we wrote any code, we also considered other approaches that have been taken to similar problems.

The autonomous vacuum cleaner Roomba[10] performs a functionally similar to our tennis ball collector, despite using different hardware to collect different objects. One of the strategies Roomba employs to traverse a room is to drive forward until it encounters a wall, and then to turn to a random direction, and repeat the process, which is known as a random walk algorithm[11]. This strategy works surprisingly well, as the random case is almost never the worst case, but more stateful algorithms might yield marginally better results.

Drawing from the aforementioned commercial examples, our algorithm will loosely be to drive forward until it either sees a ball, in which case it turns and collects it, or it encounters a wall or other obstacle, in which case it turns randomly, as per the Roomba example.

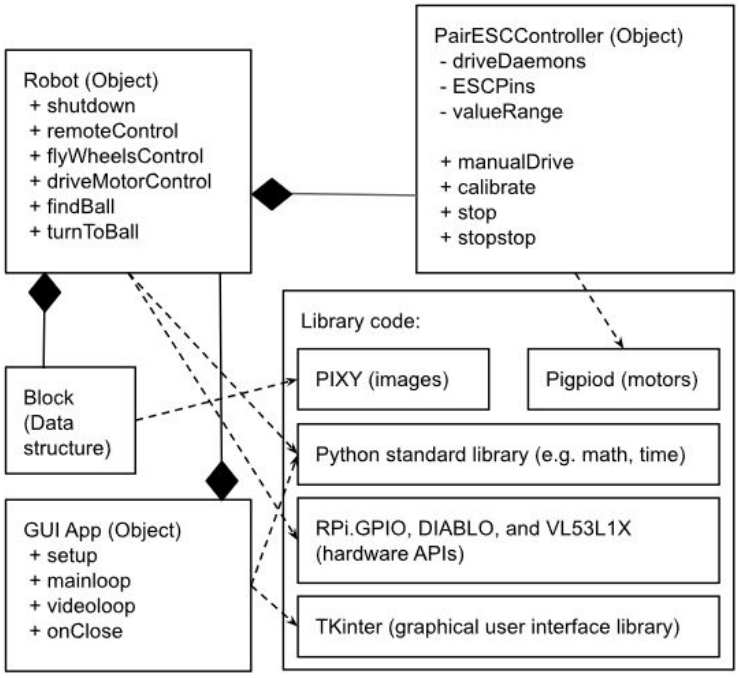
Finally, we considered the features our final software should have, which we decided should be an autonomous and remote control mode for ball collection.

[9] https://en.wikipedia.org/wiki/Decomposition_(computer_science)
[10] https://www.irobot.co.uk/
[11] https://en.wikipedia.org/wiki/Random_walk

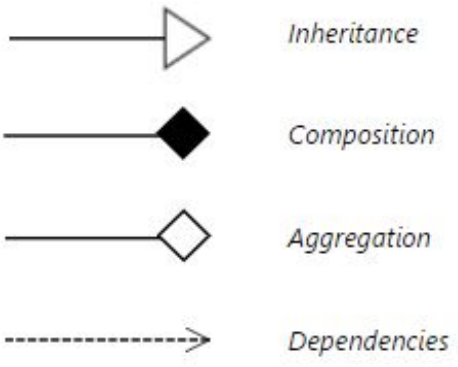
Software layout

Having considered the task, we concluded that the best solution to the problem was an object oriented model that produced an API for the functions the robot would do.

We produced a UML object diagram to plan how we would structure our code:

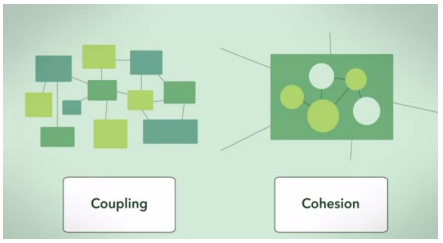


UML diagram key:



- Deriving a subclass from a parent class, retaining the parents properties, and adding new ones
- A strong association between objects, where the parent “owns” the child, and the child is only defined in the context of the parent
- A weak association between objects, where the parent “has-a” child, which has its own lifetime
- A class that is required for the function of a given class

This means that the robot is modelled as an object, which can perform functions, e.g. drive forward, and it also aggregates ("Has a"): PairESCController, which drives the flywheels; and Block, which is a data structure used to communicate with the PIXY camera. We decided that the robot would not need to store data between runtimes, so we would not need to use a database, or other non-volatile storage in our software.



This layout results in a flexible code structure, which follows software design best practices, e.g. High cohesion, low coupling[12] [13], and favouring composition over inheritance[14].

Furthermore, it conforms to the style guide provided for python, PEP 8[15], and each function has a docstring[16] (a brief comment which summarises its purpose, parameters, and return types). The full code is available on a public GIT server (https://github.com/EdmundGoodman/EES - see Version control).

These steps manifest themselves with beneficial properties, such as minimising code duplication, so it is easy to change properties, for example the pin mapped to a motor, if a GPIO pin fails. Furthermore, it is much easier to add more functionality, for example remote control testing code.

Software development

In order to ensure that our software functioned in the expected way, we tested continually throughout the development process, as much as possible (see software testing). We experienced and corrected many logical bugs and syntactical errors throughout testing, and learnt a lot about software design, image analysis and autonomous algorithm design along the way.

Controlling the output devices

The first part of the software we developed was to control the drive and flywheel motors.

For the main motors, we initially used a RPi motor controller, which provided its own software library, which we linked into methods of our robot class to allow it to drive and turn. We tested this on small DC motors, and the code appeared to function as intended. However, as discussed in the hardware section of the report, the motor controller could not supply enough

[12] https://en.wikipedia.org/wiki/Coupling_(computer_programming)
[13] https://stackoverflow.com/questions/14000762/what-does-low-in-coupling-and-high-in-cohesion-mean
[14] https://en.wikipedia.org/wiki/Composition_over_inheritance
[15] https://www.python.org/dev/peps/pep-0008/
[16] https://www.python.org/dev/peps/pep-0257/

current to drive the large windscreen wiper motors, so we discarded it, along with the code we had written to control it.

Next, we connected up a H-bridge array of relays, in series with voltage regulators to control the motors. In order to drive the motors from the relays, we had to toggle GPIO pins to switch the appropriate relays to get them to drive in the correct direction.

Finally, we switched to a motor controller which could provide enough current to drive the motors, and again provided library code for control.

Each time we changed the way we controlled the motors, we only had to change class methods in one place, due to the way we planned and designed our software, so it was not especially difficult to implement. However, there were many hardware failures, and each time we had to ensure that it was not our code which had failed, so we ended up debugging it many times.

In order to drive the flywheels, we used the python library pigpio[17], along with the linux pigpiod daemons[18]. These allowed us to interface with the flywheel motor drivers, and hence control the flywheels. We found that while driving the flywheels was fairly trivial, calibrating them was not. In order to calibrate the flywheels, we wrote a script in the “pairESCController” class that prompted the user to power cycle the flywheels at the correct time, which calibrated them.

Tennis ball analysis

Since the objective of the project is to make a machine that detects and collects tennis balls, a main facet of the software part of this project is the process of being able to recognise tennis balls. The hardware input we are using to detect tennis balls is a camera, which yields images, so we had to use image analysis techniques to achieve this.

We produced three iterations of image analysis code to try and solve this task, which were increasingly effective, and, following testing,we believe our final solution works very well.

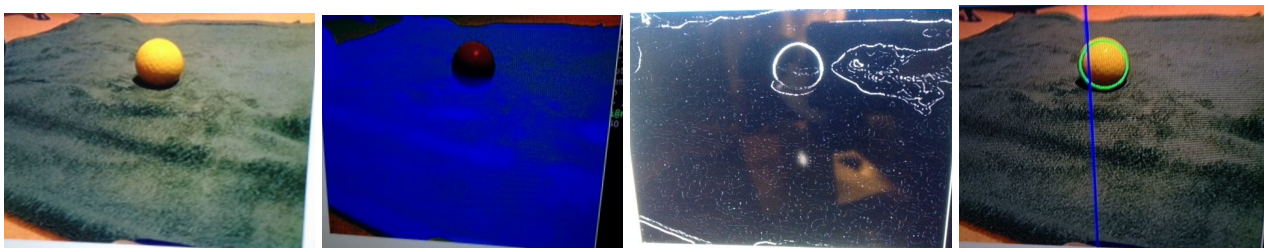
Iteration 1)

As we recognised that image analysis is a difficult problem, we started development before the robot was fully assembled, using an unmounted pi with a camera.

Our initial algorithm for recognising tennis balls in a photo was:

1) Increase the colour contrast of the image, using a CLAHE[19]
2) Apply a median blur to remove noise
3) Apply a laplacian transform to find edges
4) Blur and filter the image again to remove noise generated by the laplacian transform
5) Use the Hough circles[20] algorithm to find the circles in the image

[17] https://pypi.org/project/pigpio/
[18] http://abyz.me.uk/rpi/pigpio/pigpiod.html
[19] https://en.wikipedia.org/wiki/Adaptive_histogram_equalization#Contrast_Limited_AHE
[20] https://en.wikipedia.org/wiki/Circle_Hough_Transform



Photographs of the steps of the image analysis, left to right: Initial photo, CLAHE enhanced photo, laplacian transformed photo, Initial photo with circled detected balls

We found that this worked fairly well, however, it was initially quite slow, so we reduced the image resolution, and it produced a lot of false positives. As a result of these drawbacks, we decided to test various other analysis techniques, to see if we could find a more effective algorithm.

Iteration 2)

Next, we tried a different process for ball recognition, so we could pick the most effective one, followed the algorithm:

1) Mask to only include green pixels
2) Erode and dilate the image to remove noise
3) Find the contours in the image
4) Find the minimum enclosing circle of the largest contour

We found that this also worked fairly well, and was quite fast, however, masking for green might not work on a green coloured astroturf/tennis court, and not all tennis balls are green. Furthermore, we still found a lot of false positives occurred in testing, and looked for another approach to improve on the previous two iterations.

Iteration 3)

Following the hardware failure of our PiCamera, we purchased a PIXY2, which is a brand of camera for the Raspberry Pi which includes an onboard microprocessor. Our final recognition process was:

1) Generate an array of “blocks”
2) Write ball-like objects into each block
3) Return data about the most prominent block

We found that this worked very well, as it was faster, produced much fewer false positives, and had an easier interface for tuning, so we decided to use this iteration in our final product.

Autonomous ball collection logic

After we could control the outputs, and ascertain appropriate data from the inputs, we moved onto implementing the logic for autonomous ball collection. We started by modelling the algorithm using pseudocode:

```
LOOP
      IF a wall is closer than a metre away
            THEN turn in a random direction to avoid it
      END IF

      IF there is a ball in the camera's field of view
            THEN
                  stop driving
                  IF the ball is to the left
                        THEN turn left until the ball is central
                  ELSE IF the ball is to the right
                        THEN turn left until the ball is central
                  ELSE (the ball is no longer in view)
                        THEN continue
                  END IF

                  IF a wall is closer than a metre away
                        THEN turn in a random direction to avoid
                              the wall
                  ELSE (there are no walls ahead)
                        THEN
                              drive forward till the ball is no
                                    longer visible
                              drive forward for another short
                                    period of time
                              stop driving
                  ENDIF
      ELSE
            THEN drive forward
      END IF
END LOOP
```

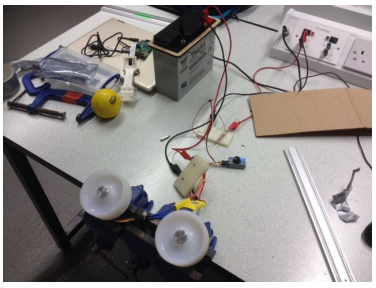
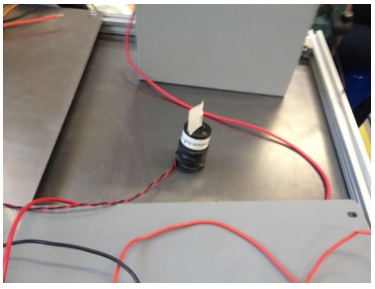
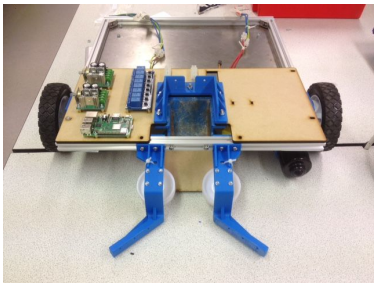
Owing to the fact that we wrote the API to our robot code in a helpful manner, it was very easy to convert this pseudocode into Python code, by replacing the abstract sentences with various predefined function calls. All that remained to do in the software component of the project was testing and tuning parameters in the logic.

Software testing

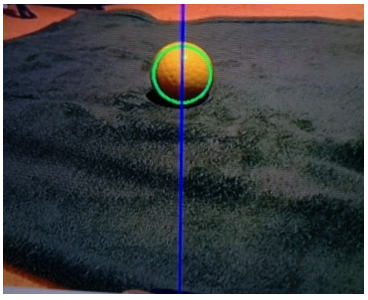
Throughout the process, we ensured we continually tested any code we had written, as, from previous experience, we know that debugging a large untested code base is a long and tiresome

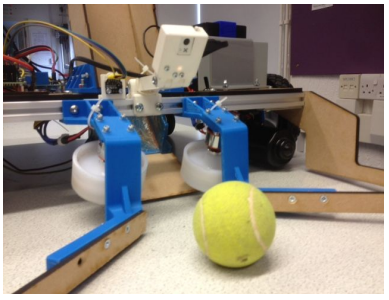
process. For each of the steps in development described above, we tested the code during and after completion.

For the code to drive the output devices, we initially tested the flywheels via a jury rigged mount, holding the motors in a pair of bench vises (1), as we had not mounted them to the robot yet, and the motor code by turning a small DC motor (2), instead of the very high current windscreen wiper motors, as our first motor driver malfunctioned. We also tested it again when everything was mounted to the robot (3).

For the image recognition, we tested each algorithm by showing the camera tennis balls, and verifying they were detected, along with testing if it was shown an empty floor, it didn't detect any balls. We also wrote a script that displayed the process of recognition, with each step being shown as an image on the screen, and the final detected ball being circled on the input image (1). In early iterations, we noted that there were many false positives, perhaps most worryingly the robot classified feet as balls (2), so we changed the algorithm. We finally finished development with an algorithm which worked very fast, and produced few false positives (3).





For the autonomous control logic, we tested it on the robot, as at that point it was mostly assembled. We initially tested it inside, but quickly moved to testing outside, as there was not enough ambient lighting, and early iterations of the control logic often ended up crashing into table legs, and the hardware design team was none to pleased with fixing any damage incurred.

Deliverables

We produced a program that autonomously controls the robot to pick up balls on the tennis court, using the aforementioned image recognition algorithms. Initially, the program ran on a CLI

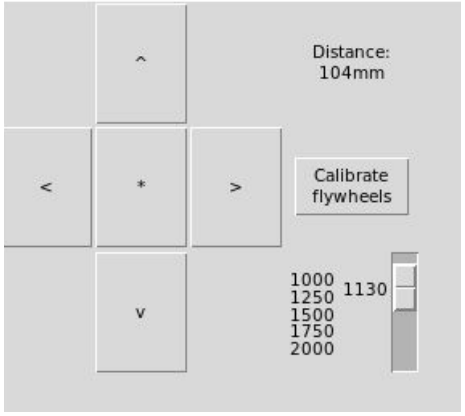
(command line interface), from the terminal, but in order to improve usability for the end user, we also produced a GUI (pictured below) to allow manual control of the robot.

The remote control GUI has the main control buttons to maneuver the robot by driving forward & backward, and turning left & right. There is also a center button that toggles the flywheels on and off, and lights up red when on, as a safety feature, as the flywheels could be dangerous if the operator did not know they were active. Furthermore, there is a readout which displays the distance to the wall detected by the time of flight sensor in real time, a button to start calibration of the flywheel motors, and a slider to control the duty of the flywheels, to allow tuning for different types of tennis balls and surfaces, which defaults to 1130, which we found to work well during testing.

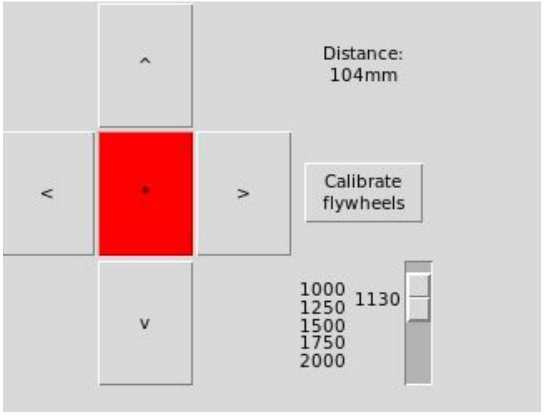
The autonomous GUI has a button to start collection, and again buttons to calibrate and set the duty of the flywheels.

The CLI allows both remote control and autonomous function, and each mode is entered by typing the character in square brackets after each option.

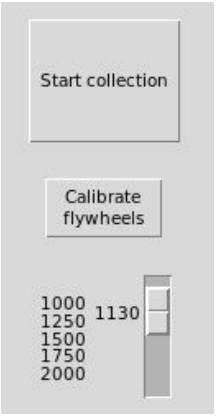
See the user guide for more information.



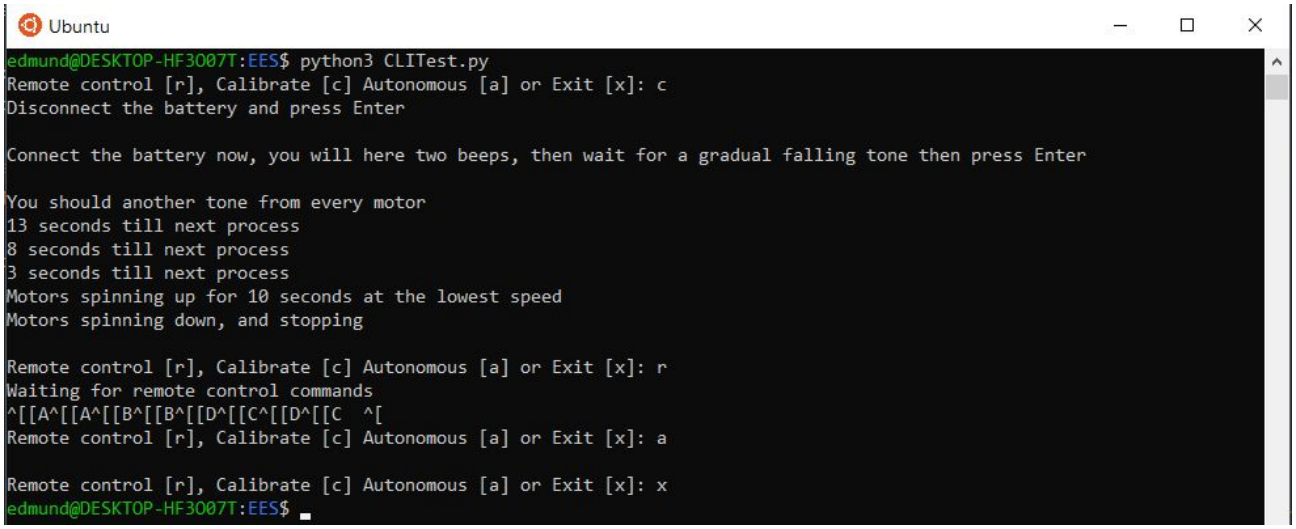
The manual GUI with the flywheels off



The manual GUI with the flywheels on



The autonomous GUI



The CLI, with each setting having been chosen

Version control

Throughout the software development process, we used Git version control, which is a software package that affords various features, such as:

- The ability store previous versions of our code, so you can could roll back any changes, for example accidentally deleting an important file
- The ability to share code between computers offline, by pushing and pulling to a server, which means that you can write code without having to be near the robot
- The ability to merge multiple versions of code together, which allows multiple developers to collaborate on writing one file, without them breaking each others changes
- To allow us to publish the code easily as a public repository on GitHub[21] - https://github.com/EdmundGoodman/EES, if you follow this link, it will take you to a website where you can view all of our source code, and see changes as they are committed to the server

In software development, good version control is immensely important in any large scale project, for all of the aforementioned reasons, and our use of it saved us from a lot of work various times.

[21] https://github.com/

The work tree (a list of all the changes made to the source code) for the git server at time of writing is in appendix 2: Version control tree.

Summary

In summary, we produced a software package that fulfilled all of product requirements, including an easy to use manual control package, and an autonomous control option, that detects and collects tennis balls very effectively. In order to achieve this, we used a vast array of advanced techniques, including:

- **Object orientation** of the robot, the ESCs, and the GUI app
- **Parallel processing**, using the PIXY2 for image analysis, and the Raspberry Pi for autonomous control simultaneously
- **Image analysis techniques**, such as masking, blurring, eroding, dilating, and more specific techniques, such as hough circles, and minimum enclosing circles.
- **Multi-threading** to allow real time updates of distances in the remote control GUI
- **Lambda functions**, to map GUI events to robot functions
- **Procedural, functional, and data abstraction** of robot functions
- **Interfaces between languages**, including reading PIXY2 C++ data into the robot Python code, and using Python to drive the bash pigpio daemons
- **Context managers**, to allow entering and exiting the CLI remote control code

We believe that the software we produced fulfills the product requirements, and that it would be easy to maintain/add new features to in future.

Environmental Considerations

Battery considerations

We decided to power our robot using a 22Ah sealed lead-acid battery, and a 2200mAh LiPo power bank (see discussion of battery technology in development section).

Although the lead-acid battery contains hazardous concentrated sulphuric acid, it is sealed in a strong plastic case, so it is very unlikely to spill, unless significantly damaged, however, it is fairly power efficient under load, due to its low internal resistance, which facilitates high current draw. As a result of this, it is fairly environmentally friendly, if it is disposed of properly after use.

Similarly, the LiPo power bank contains lithium metal, and due to its high energy to volume ratio and inherent chemistry, it can be liable to catastrophic failure, including deflagration or explosion. However, it is also enclosed in a sturdy plastic case, which similarly mitigates this risk, and is quite environmentally friendly compared to other less efficient batteries, if disposed of correctly.

Carbon footprint, from energy calculations on the batteries

Considering the lead acid battery[22] (22Ah, 12V):

Finding the total energy stored:

$E = VIt = 12 \times 22 \times 3600 = 950400J$

However, if you use all of the energy stored in some types of battery, including lead acid batteries, they degrade. An estimate safe value for to draw from a lead acid battery is 80% of its total stored energy, so the viable energy to use is:

$E' = E \times 0.8 = 760320J$

Furthermore, the rated capacity (Ah value) of a lead acid battery is normally given for a 20 hour discharge rate, but if you have a high discharge rate, the capacity falls steeply. A common rule of thumb is that for a 1 hour discharge, which in our case is a steady current of 22A, you only get half the rated capacity. Since we are running 2 windscreen wiper motors, and 2 brushless outrunner motors in series, we are likely drawing an average current of:

$I = 2 \times 5 + 2 \times 7 = 24A$

Since $24 \approx 22$, we can estimate to half the rated capacity, so we get a final stored energy of:

$E'' = E' \times 0.5 = 380160J$, which is dissipated over about 1 hour

We can then also consider the LiPo power bank (2.2Ah, 5V @ 2A) in the same way:

$E = VIt = 5 * 2.2 * 3600 = 39600$

$E' = E \times 0.8 = 31680$

$E'' \approx E' = 31680$, since $2.2Ah \approx 2Ah$

Therefore our total energy consumption per hour of runtime, due to charging the batteries is:

$E_{total} = 31680 + 760320 = 792000J$

We can then calculate the estimated mass of CO_2 released owing to our robot, per charge cycle.

Assuming we are using a fossil fuel based generation of electricity from the national grid, we can estimate that for 1 kWh of electricity, 537g of CO_2 are produced[23]. Since $792000J \approx 0.22kWh$, per hour runtime, our robot will produce 118g of CO_2 for each 1 hour battery cycle, which is about 4 sweeps of the pitch.

Considered solar panels

We considered using solar panels in our design to make it more environmentally friendly, however we concluded there was not a good way to mount them without being damaged, and they did not produce very much power compared to our very high current motors, so the effect they would have on charging the battery would be negligible. A possible solution to this would be to build a separate charging station with solar panels built in which charged throughout the day however we considered such a construction beyond the scope of this project.

[22] https://www.powerstream.com/battery-capacity-calculations.htm

[23] http://www.knowlton.org.uk/wp-content/files/Energy%20carbon%20conversions.pdf

Regenerative braking

The motor driver we used in our final product, the PiBorg Diablo, features regenerative braking, so we can recharge our batteries when we brake or change direction, which will increase the environmental friendliness of our robot as less energy is wasted while driving.

Materials and design processes

- Additive manufacturing (3D printing) was employed for most of the mounting brackets and odd shaped components. This meant the only waste was support material which we tried to minimise by designing the parts to print without supports where possible
- Larger components were produced from MDF using a laser cutter. Since this is a wastage method, we decided to use MDF rather than acrylic as it is made from recycled wood fibres and recently new recycling methods[24] have been developed that should help reduce the amount of waste sent to landfill or incinerators. Additionally, unused section on MDF sheets can still be used for cutting other small parts at our school.
- If we were to mass produce this design, we would likely injection mould or vacuum form most of the components to improve rate of production however for a one off prototype it made sense to use these methods instead.

Team Working

Meet the team

Daniel - Project manager, Software, Lead electronic design
Martin - Lead hardware, Lead design & aesthetics
Edmund - Lead software, hardware
Freddie - Hardware
Ruben - Hardware
Athena - Research, design & aesthetics

[24] https://www.edie.net/library/Test-bed-Scaling-up-a-world-first-recycling-solution-for-MDF/6764

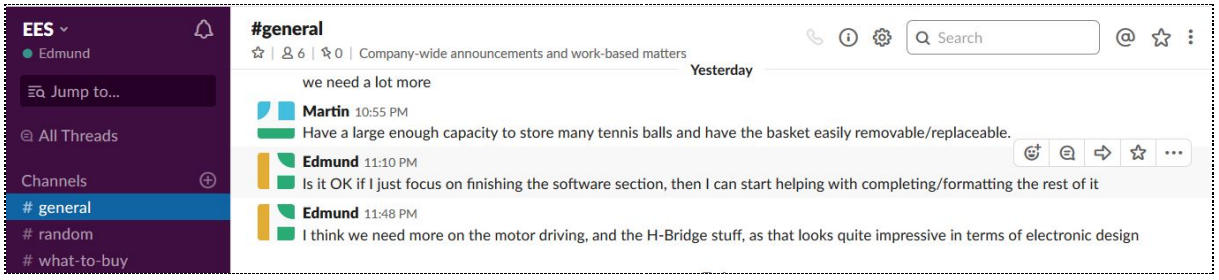
Ability to contribute time to the project

Some members of the team had more commitments than others, so were able to contribute less time to the projects. As a result, we allocated tasks accordingly, with those members who were able to contribute more time being given lead roles.

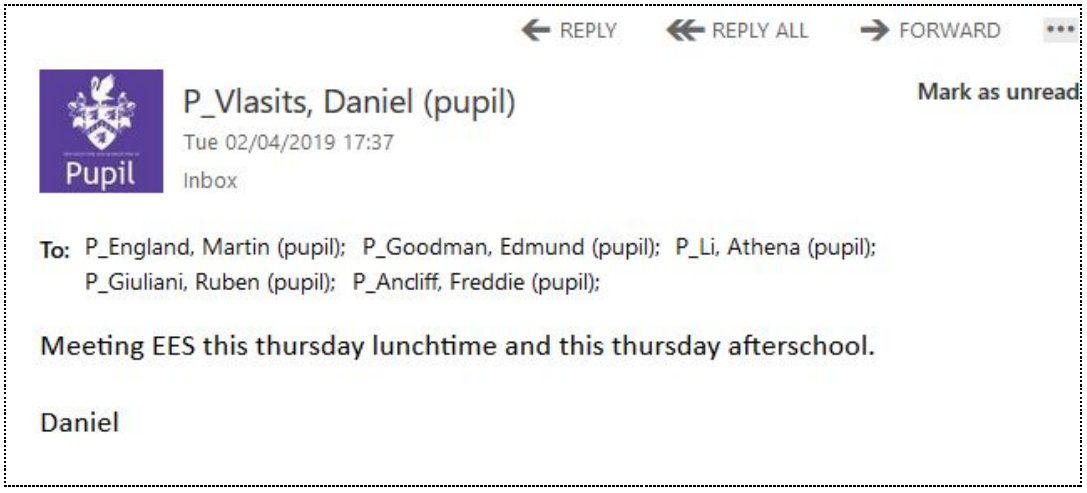
Communication, Collaboration & Conflict resolution

In order to ensure we communicated effectively throughout the process, we had regular team meetings, every Wednesday after school, and occasionally scheduled extra meetings in order to achieve objectives in a timely manner.

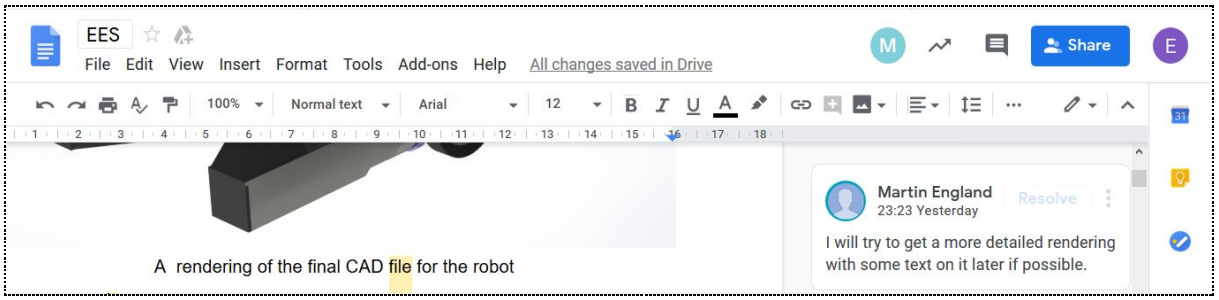
Furthermore, we also used various tools, including email and slack for communication, and Google Drive for collaboration.



A screenshot of a conversation between team members on slack

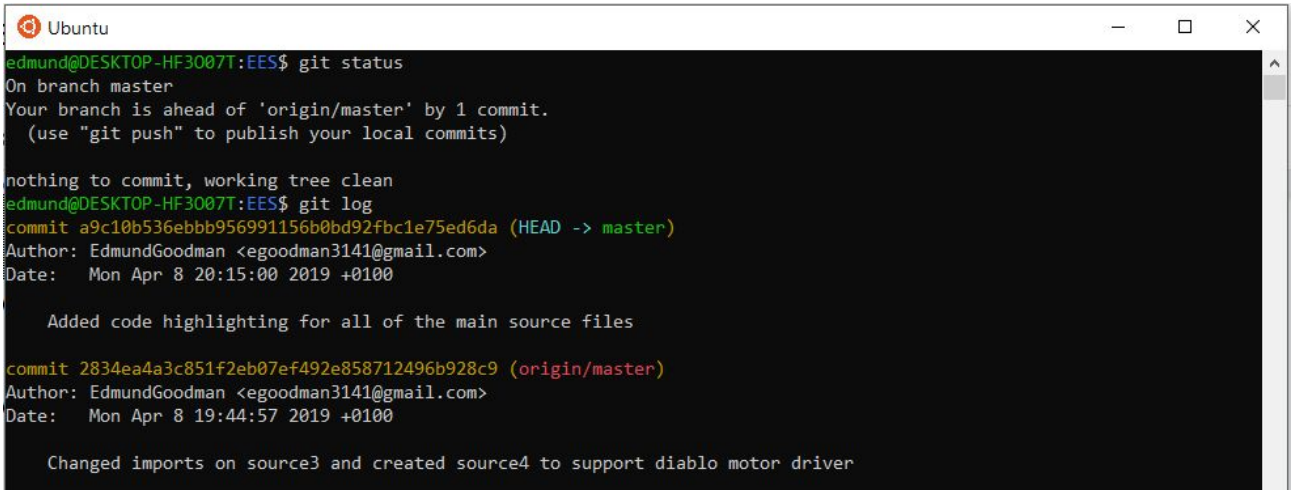


A screenshot of an email organising an extra meeting



A screenshot of our google drive document for the report, with a comment on

Furthermore, we used a GIT server during software development (see version control), which allowed us to collaborate between various machines, including our production machine.



A screenshot of the command line interface to the Git server

We did not experience any major conflicts during the development process, and hence didn't have to do much conflict resolution. However, this lack of conflict was most likely due to strong communication between team members, in the ways detailed above.

Task Allocation

During each meeting on Wednesday after school, we allocated tasks for team members to complete for next week, or to work on for an extended period of time. For example, during the analysis and planning stage of the project, we each produced a powerpoint detailing the options we could consider for a specific piece of the project, for example: which battery to use, which motors to use, which control board to use, which ball collection system to use, etc. The next time we met, we all presented our slides, and made the decision on which item to use. Later on, during the development phase of the project, we set short and long term goals, e.g. mount the box to the robot for next week, or get the image analysis working over the next few months.

This task allocation procedure allowed us to effectively use time, and not fall behind with our initial time plan (see gantt chart)

Conclusion & Evaluation

The final product of this process is an autonomous and manually controllable tennis ball collecting report. Pictures of the product are shown below:

The robot is able to collect tennis balls by randomly traversing the court and searching for tennis balls through a camera and collecting those that are detected. It can also be controlled manually by means of a simple Graphical User Interface. Tennis balls are collected by two rotating flywheels at the front of the robot which project the ball up a ramp into a removable storage basket at the back of the robot.

Comparison to Specification

Overall, the final robot meets the points of the initial specification well. The original specification is compared to the final robot below.

- Aesthetics
  - ***The robot should not be intimidating by moving very quickly or making very loud noises.***
  - ***The robot should look neat and professional.***
  - The flywheels are enclosed by shell plates so they do not pose a risk to public safety. Furthermore, the flywheels are not activated constantly only when a ball is detected to save battery and also to reduce noise.
  - The low geared windscreen wiper motors which are used to drive the robot restrict the robot speed so that it is not intimidating when collecting.
  - The bulk of the robot is enclosed by shell plates which cover wiring and electronics giving the robot a tidy and professional appearance.
- Customer
  - ***The robot should be easy to control.***
  - The robot can be manually controlled using an intuitive GUI (graphical user interface). The interface is not excessively complex and buttons are laid out in a

logical manner and labelled so that it is very easy for any user to operate with little instruction.

- Cost

Items required for final iteration

Item	Quantity	Cost for 1 item	Total cost
Raspberry 3B+	1	£32.38	£32.38
12V 22AH Lead Acid Battery	1	£43.99	£43.99
Windscreen Wiper Motors	2	£17.00	£34.00
4x 75mm heavy duty rubber swivel castor wheels	1	£6.39	£6.39
2020 Extrusion 500mm length	4	£6.69	£26.76
pi heatsink 3pcs	1	£3.39	£3.39
Sandisk micro SD card + adapter	1	£5.99	£5.99
VL53L1X Time of Flight (ToF) Sensor	1	£9.46	£9.46
1000kV outrunner brushless motors + 30A ESC	2	£13.64	£27.28
wheels	2	£2.99	£5.98
T nuts for aluminium extrusion	1	£8.29	£8.29
Diablo Motor Controller	1	£66.66	£66.66
Pixy2 camera	1	£55.99	£55.99

    - ***The robot should be within budget and the final cost should be less than £400 (from customer surveys)***
    - Our method of finding and collecting tennis balls used image analysis to locate them relative to the robot. This was a very difficult task in terms of software and hardware and as such the prototyping costs were high. However, the total cost for the table above showing the final items used comes to £326.16 and therefore does fall into the price range stated above when sold for profit
- Environment
    - ***The robot should be weatherproof and able to operate in different weather conditions.***
    - ***The robot should have a rechargeable battery so that it can be used repetitively and parts should be recyclable if possible.***
    - All electronics are covered by weatherproof shell plates which protect the robot from rain. The rubber tyres are able to grip the court in high winds or wet conditions.
    - The lead acid battery which powers the robot has a very long life and many charge cycles. Furthermore, the body of the robot is made from stainless steel plates, MDF boards and aluminium extrusion. All three of these materials can easily be reused or recycled.

- Size
    - ***The robot must be large enough to carry a large number of tennis balls.***
    - The removable basket that sits on the back of the robot is large enough to hold roughly 30 tennis balls. Usually no more than this would be on a court at once. Furthermore taller baskets can be fitted which can hold almost twice as many tennis balls.
- Safety
    - ***The robot must be safe to operate in a public area with no exposed wires and ability to detect and avoid people.***
    - Waterproof shell plates cover the electronics so there are no exposed wires which can be touched accidentally by the public.
    - Time of flight laser range finders are mounted to the front of the robot which can detect incoming obstacles (be that people or walls). The robot is then programmed to take action to avoid these obstacles by stopping and turning until an obstacle is no longer detected then driving away.
- Function
    - ***The robot must be able to autonomously collect and store tennis balls.***
    - Collection is achieved by two counter-rotating flywheels which launch the ball into a storage box at the back of the robot.
    - Tennis balls are detected by a front facing camera and recognised at a software level. The robot then turns towards the ball, spins the flywheels and drives straight towards it.
- Material
    - ***The robot must be made from a strong and durable material which is weatherproof and can support the weight of the tennis balls and components.***
    - The main frame of the robot is made from aluminium box section which is reinforced with stainless steel plates therefore the frame is very rigid and strong.
    - The ramp which directs the tennis balls into the hopper is 3D printed and reinforced with fibreglass to improve impact resistance.
    - The robot is also covered with shell plates which are coated with a weatherproof paint. However, there are still a few gaps in places which would need to be sealed to make it fully weatherproof. Sealing these gaps would cause more problems than solutions because repairs to the electronics would be very difficult.

Further Development Options

- One option which was also highlighted in our survey was to adapt the robot to collect hockey balls as well as tennis balls. However, hockey balls and tennis balls have different diameters so cannot be collected at the same time and the flywheel spacing would have to be adjusted manually. In addition, hockey balls are much heavier and harder so would increase the risk of breakages on the robot.

- Another option is to adapt the robot to collect rubbish in public spaces such as drinks cans, this would be a more technically difficult challenge however, it would make the product useful for almost any company. On the contrary, collecting rubbish would be easier with a robotic arm rather than flywheels due to the irregular shapes and sizes and also would be

difficult to detect using image analysis. This is again due to the fact that rubbish has irregular shapes and sizes and colours; all these features are useful in detecting the presence of an object in an image.

- Another feature which could be added is a small GPS unit which would allow the robot to record where it has previously been so it does not repeatedly sweep the same area therefore the collection would be much more efficient.

- A camera which could rotate (or two cameras facing in opposite directions) could be mounted to the court nets. These would be able to communicate the location of more balls to the robot (via bluetooth or wireless connection) so that it could plot a more time-efficient course to collect all the balls.

- A separate feature to the robot would be a charging station which the robot would automatically return to after collecting all the balls on the court. The robot would then be recharged there. Solar panels could also be added to the charging station which would extensively reduce the environmental impact of the robot.

User Guide

Turning the robot on

The user should press the on button on the power bank to power on the pi, and then wait about 20 seconds till the pi has booted up, which is shown by a constant red power light on the circuit board, and the green light adjacent to it should be off

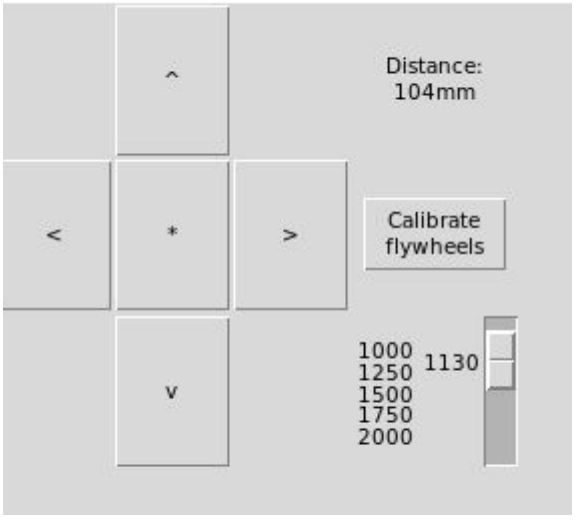
The user should then flick the main battery switch to power on the drive and flywheel motors.

The user can then VNC or SSH into the pi, and start the program they require, e.g. the CLI code, or the remote control GUI.

Calibrating the flywheels

Initially, select the option to calibrate the flywheels, in the GUI or the CLI, then follow the instructions printed to the terminal, which are duplicated here:

1. Disconnect the battery, and press enter
2. Reconnect the battery, wait for two bleeps and a gradual falling tone from each motor, then press enter immediately
3. You should then hear another tone, as automatically completes its calibration sequence, taking about 25 seconds, in which it requires no other intervention

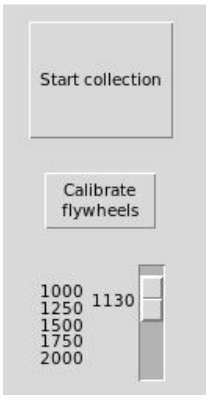


Using the remote control GUI

The four arrow buttons control the direction of movement, with “^” driving forward, “>” right, “v” backward, and “<” left.

The center “*” button toggles the flywheels on and off, turning red when on, and the slider sets their speed (this should normally be quite low, as too high will start vibrating and damaging the robot, and shoot the tennis ball far over the bucket). However, the flywheels must be calibrated by pushing the calibrate flywheel button after powering on, before they can be turned on.

The distance text displays the current distance to a solid obstacle in mm, so the user can avoid it.

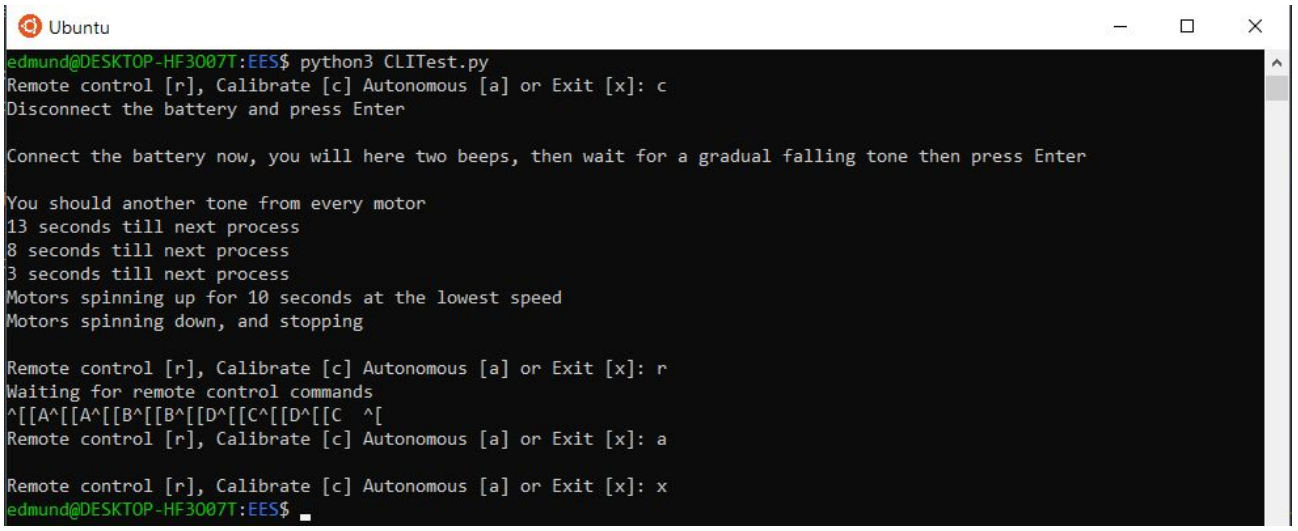


Using the autonomous control GUI

As before, the flywheels must be calibrated before use, and this can be done by hitting the calibrate flywheels button. Similarly, the speed of the flywheels can be set with the slider, and again should be quite low for best function.

In order to start autonomous collection, press start collection, and close the window in order to stop it.

Using the CLI



In order to use the CLI, the user enters the letter in square brackets following option they wish to select.

In order to calibrate the motors, the user should follow the printed instructions, which are duplicated previously in the user guide

In order to remote control the robot, the user should use arrow keys to drive, the space bar to toggle the flywheels on and off, and tab to print the distance to the nearest obstacle in millimeters, these will display control characters to the terminal e.g. “^[[A” for up as above, which can be ignored by the user

Webliography

1. https://www.yankodesign.com/2014/07/30/the-ball-boy/
2. https://www.yankodesign.com/about/
3. https://motherboard.vice.com/en_us/article/8qx8v4/will-tomorrows-ball-boys-and-girls-be-robots
4. http://courses.me.berkeley.edu/ME102B/Past_Proj/f09/1%20BearClaw%20Tennis%20Ball%20Collector/index.php
5. http://courses.me.berkeley.edu/ME102B/index.html
6. http://courses.me.berkeley.edu/ME102B/Past_Proj/f09/1%20BearClaw%20Tennis%20Ball%20Collector/code.txt
7. https://www.payscale.com/research/US/Job=Tennis_Instructor/Hourly_Rate
8. https://www.raspberrypi.org/downloads/raspbian/
9. https://en.wikipedia.org/wiki/Decomposition_(computer_science)
10. https://www.irobot.co.uk/
11. https://en.wikipedia.org/wiki/Random_walk
12. https://en.wikipedia.org/wiki/Coupling_(computer_programming) [visited 4/4/2019]
13. https://stackoverflow.com/questions/14000762/what-does-low-in-coupling-and-high-in-cohesion-mean
14. https://en.wikipedia.org/wiki/Composition_over_inheritance
15. https://www.python.org/dev/peps/pep-0008/
16. https://www.python.org/dev/peps/pep-0257/
17. https://pypi.org/project/pigpio/
18. http://abyz.me.uk/rpi/pigpio/pigpiod.html
19. https://en.wikipedia.org/wiki/Adaptive_histogram_equalization#Contrast_Limited_AHE
20. https://en.wikipedia.org/wiki/Circle_Hough_Transform
21. https://github.com/
22. https://www.powerstream.com/battery-capacity-calculations.htm

23. http://www.knowlton.org.uk/wp-content/files/Energy%20carbon%20conversions.pdf
24. https://www.edie.net/library/Test-bed-Scaling-up-a-world-first-recycling-solution-for-MDF/6764

Gallery

Appendices

1: Links to budgeted components

Item	Link
Raspberry Pi 3B+	https://amzn.to/2YYTPjw
12V 22AH Lead acid battery	https://bit.ly/2U37ouE
Windscreen wiper motors	https://bit.ly/2OYGvHc
4x 75mm Heavy duty rubber swivel castor wheels	https://amzn.to/2uS58Mw
RPi, motor driver board MC33886	https://amzn.to/2Z0k1dt
2020 Aluminium extrusion 500mm length	https://amzn.to/2UEYiIT
Pi heatsink 3pcs	https://amzn.to/2G6Zprg
Sandisk micro SD card + adapter	https://amzn.to/2UH9CUN
Pi camera	https://amzn.to/2KiIZkT
Pi camera extender	https://amzn.to/2GbqNpm
VL53L1X Time of Flight (ToF) Sensor	https://bit.ly/2uWD5eZ
1000kV Outrunner brushless motors + 30A ESC	https://amzn.to/2Z0ly3d
Rubber wheel 150mm diameter	https://bit.ly/2Kmo8NQ
T-nuts for aluminium extrusion	https://amzn.to/2VBly7K
Diablo Motor Controller	https://bit.ly/2ULxZk9
Pixy2 camera	https://bit.ly/2KneuKH

2: Version control tree

```
* 2834ea4 (HEAD -> master, origin/master) Changed imports on source3 and created source4
to support diablo motor driver
* f1a85c9 Updated turning logic, added docstrings to ESCD3in and tkinterTest, and moved
some old testing files
* d496e64 Modified GUI to remove testing code
```

```
* 03ea8f4 Improved the GUI interface to control the robot
* bb57a6a Fixed tkinter GUI, changed default duty, and added more autonomous logic,
following testing
* 46a5cdf Fixed bugs during testing
* 80da302 Merged changes onto PI
* 8d0f6b0 Major directory schema change, replace danielCode with useGitEESDanielcode
* 241f723 Added html file containing the colourised git tree
* c038b50 Added code to do 'Roomba' strategy, randomly turning when encountering walls
*   dec2607 Merged code from PI and local machine
|\
| * ecae908 Added testing code
* | 36b6d75 Merged turning logic into main source file
* | 61f8a2c Changed turning logic
|/
* 08c8ec7 Restructure file tree, moved pixy files into main directory, and added random
turning, and docstrings to source3
* f710078 Merged code from DanielCode into source3.py
*   63b8391 Merged files into main tree
|\
| * 4e18052 Added PIXY image rec code
* | f6c3e40 Modified the GUI
* | 6123bb4 Added GUI for controlling robot, added changed copy of source2 to prevent
merge conflicts, simplified ESCD3in
|/
* daa8698 Added changes
* 2a6c1f0 Dan fixed some code
* 725a2ae Added daniel's code from SD
* faff018 Added some testing code
* 942e103 Bug fix in scope of img analysis variables
*   af3a80c Merge branch 'master' of https://github.com/EdmundGoodman/EES
|\
| * 5dda5c9 Changed ball collection code by testing
* | 316975a Added camera testing code
|/
* cf83584 Added new testing code
* 505dbd0 Updated turning logic
* 2e91910 Improved turning code
* e423572 Added turing code
* 18a1963 Update
* 4f169f1 Changed turning logic
* 2c17410 Removed unnecessary files'
* 04860e5 Changed to only include sleep from time
* b4fdd6b Paramertised image analysis code
* 0b4233b fixed turning code
* 6cf0826 Test
* 3b98c9d Added image rec code
* 9ec464d Added debugging code
* 97dac3d removed file
* 9b797b8 Added functionality code
* 99d8f4f Updated driving, and added flywheel code
*   fb3fb31 Merge branch 'master' of https://github.com/EdmundGoodman/EES
```

```
|\
| * 1da6c3c Modified code to allow running on real pi hardware
| * 7944d54 Added code to turn towards ball
| * 59a6ebd Updated ball test low res
| *   dc2264d Merge branch 'master' of https://github.com/EdmundGoodman/EES
| |\
* | | 113c6ae Added Daniel's testing files
* | |   082d128 Merge branch 'master' of https://github.com/EdmundGoodman/EES
|\ \ \
| |/ /
|/| /
| |/
| * 2777e7a Added saving of photos, and increased speed of analysis from .5s to .3s
* |   2b1f982 Merge branch 'master' of https://github.com/EdmundGoodman/EES Necessitated
to merge local copy of development with main server test case upload
|\ \
| |/
| * 588b5d2 Added test code
| * 2efba59 Added camera interface on pi
* | 019b4ea Added old control code for ease of testing
|/
* 49ff688 Added image analysis software & test cases
* 9f57a4e Added functioning drive for main wheels
* a4f1319 Added source.py and updated project description in README.md
* 8353684 first commit
```