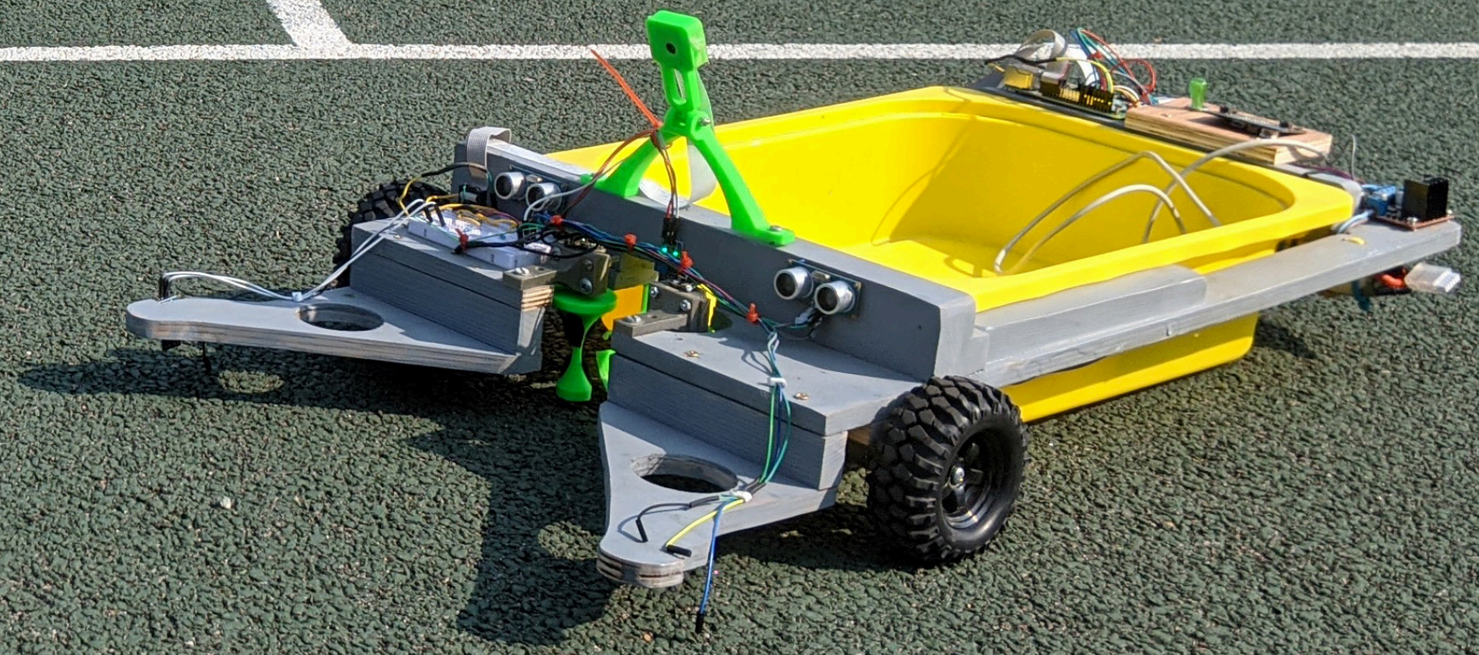
CREST

TENNIS-SCARAB

REPORT



By William Woodard

Page of Contents

This is my submission for a Crest award. It documents the process of designing, making, coding and testing the 'tennis scarab': A robot designed to automate the job of a ball boy by going around a court and picking up tennis balls. This is to allow small clubs and individual players to spend more time on the sport but also to enable those who physically cannot collect the balls, to still play tennis.

Project brief

Introduction

Tennis is an extremely popular sport. At a club and hobby level, players have to pick their balls in their own time. This takes ages and often means they must sacrifice game time for a chore. For safety reasons balls need to be collected off court after each game. This means some people are unable to play tennis because of accessibility issues. The solution to this problem is an autonomous tennis ball collecting robot.

Requirements

The robot must be:

- Cheap and easy to produce so everyone can access it
- Open source
- Intuitive to start
- Portable
- Aware of its surroundings to it does not bump into the net or get in the way of a game
- Fast
- Able to act autonomously so players can get on with there game
- Waterproof
- Able to store a lot of balls and then act as a mobile ball bin
    - The average set will leave 6 balls discarded so the robot must be able to collect these and store them while the next set is played
- Able to operate for at least an hour

Plan

Look at previous solutions

Come up with as many designs as possible

Pick the best design

Develop it further

Estimate cost and pick parts

Evaluate, Design and make a prototype

Test prototype

Further develop design

Code the autonomous part

Optimise the prototype

Risks and dangers

Risks to robot users

- Li-Ion battery blows up
    - Make a protective case around the lithium ion battery to stop it from rupturing
    - When charging, keep in a lipo-safe bag
- Could cause tennis player to slip up
    - Have some sort of ‘foot-detection’ method to stop robot from going near players
    - Have it only patrol while players are having a break, out of the way
- Electric shock
    - Have no exposed wiring

Risks to me while making the robot

- I might get burnt while using a soldering iron
    - Be careful and wear gloves while using it
- If I decide to use metal shell, I will have to do some welding which presents lots of risks
    - Do this under instruction of a trained professional
- I could get an electric shock while doing the wiring
    - Turn off all appliances while I am doing the wiring
    - Do not use high voltages

Previous solutions

Tennibot

This is an actual product that you can buy so it has gone through a lot of testing and optimization however it is very expensive. I plan on making my solution a lot cheaper. The things I like most about this design are the portability of the shell, and the way it picks up tennis balls. The way the shell is designed, you can pick the whole robot up and drag it along on wheels. You can also take the tennis ball storage bit out. The way it picks up tennis balls is it has 2 wheels spinning around fast. As these are perfect positioned so the space between them just about fits a tennis ball, they will propel the tennis ball up a slide into the basket (See diagram below). As tennis balls are slightly squishy, the gap between them could be slightly too small so it has more grip on the tennis ball. The one disadvantage to this solution it the fact that it can only process 1 ball at a time. This slows it down a bit however it has 2 ramp like things on the front to position the tennis balls in the correct position to get propelled and also it acts as a space for a buildup of tennis balls. Another cool thing that the tennibot has is a transponder which you clip to the net. This allows the robot to act autonomously as it know how far it is away from the net and there for wont go too far away from or too close to the net.

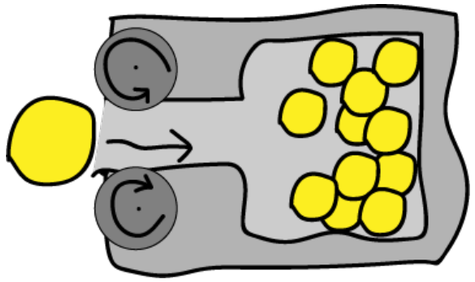


Photo: Tennibot.com Illustration: I drew this on illustrator

Inwatec lawnmower

A company called Inwatec converted an old lawnmower into a tennis ball collector for their local tennis club. The way they did this is they took an old lawnmower (Bosch AHM38G) then took off the metal blades used to cut grass and replaced them with 2 softer ‘blades’. These, when rotated fast, pushed the tennis balls into a box very quickly. There solution had a big advantage in that it could collect multiple tennis balls at once. However their solution was not autonomous and was powered by a human pushing it around. If I use a similar solution, I will employ a motor to turn the blades and a robot to push it all around. I will also use a camera to identify the tennis balls.

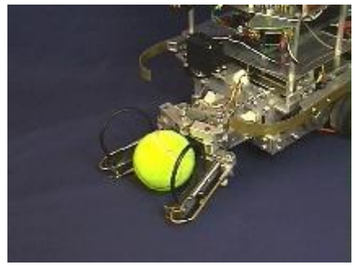
Photo: youtube.com

Previous solutions

Robot arm

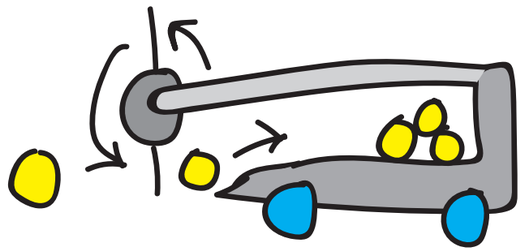
One team made a tennis ball collector using a robot arm. However, this solution was very slow and difficult to get working. It would also be expensive to build.

Source: https://newtonlabs.com/pictures/m1-grab-2-33.jpg



Brainstorming different methods to capture tennis balls

Lawnmower method



Multiple balls can go in at once

Might use lots of power

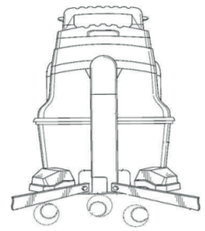
Simple

Fast

Inspired by inwatechs lawnmower method,
a rotating axis with 2 blades on it push balls into storage

Hoover method

A hoover sucks the balls into place



Will also pick up dirt

Uses lots of energy

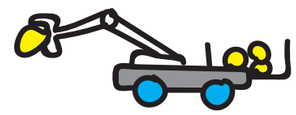
Robot arm

A robot arm picks up tennis balls
and puts them into box

Complicated

Expensive

Slow

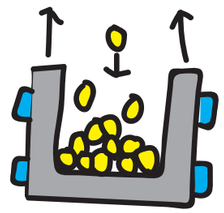


‘Push method’

Very simple

Easy and cheap to make

Means the robot cant go backwards



Fast

Just pushing the balls

Flywheels

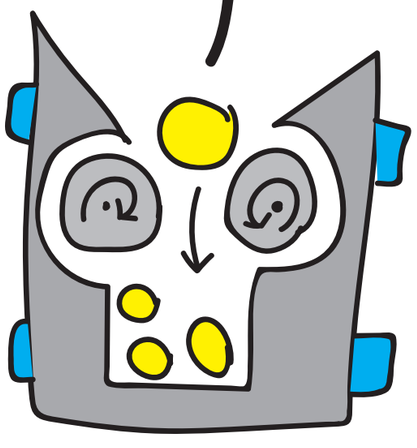
Can only process 1 ball at a time

Simple

Might use lots of power

Fast

2 spinning flywheels propel ball into storage



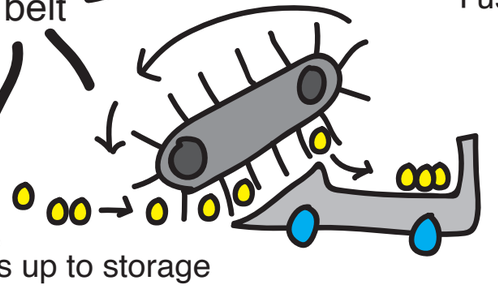
Conveyer belt

Slow

Heavy

Complicated

Lots of moving parts

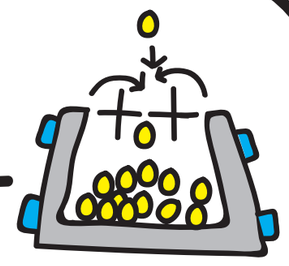


A conveyer belt brings balls up to storage

Rotating crosses

Simple

Easy to make



Slow

2 rotating crosses push balls in
and stop them coming out

Gate method

Simple

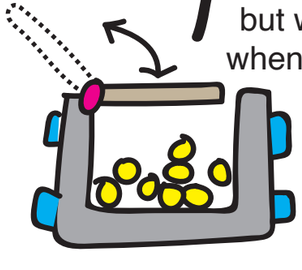
Fast

Easy to make

Only 1 ball can go through at a time

I used a similar design before
so I have experience

Similar to push method
but with gate which closes
when robot goes backwards



Deciding what design to use

I originally decided to go with the gate option and started designing the chassis. When looking at the dimensions though, it occurred to me that this option might not work as the resistance from all the balls would be too much for the motor to overcome and it would not be able to move. Also, the user would not be able to remove the balls without picking them up one by one. Because of all this I decided to go with the flywheel option: The balls will be propelled by some spinners, up a ramp into a tray. This was very simple, fast and relatively easy to make. Also, I could put the balls into a tray in the center of it which a user could remove before putting an empty one in.

This is how I started the chassis design; I found an old ikea tray and measured its dimensions before adding 3 cm on either side, 5 cm on the back for the electronics and then a large front for the spinner contraption. I then went on to designing all the 3d printed parts for the spinner contraption.

Deciding what sensors to use

Ultrasonic sensor

When the robot is patrolling around the court it will need to turn when it reaches the outside net. To sense this I could use a distance sensor.

Camera

I will use a camera to do things like tennis ball detection and foot detection so it can avoid going into people. I might also be able to use this to detect the outside court netting.

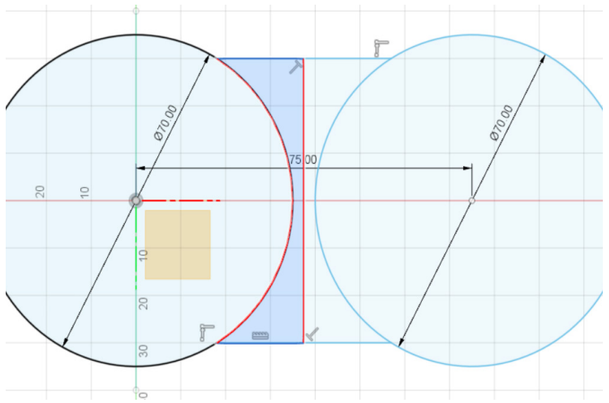
IR line sensor

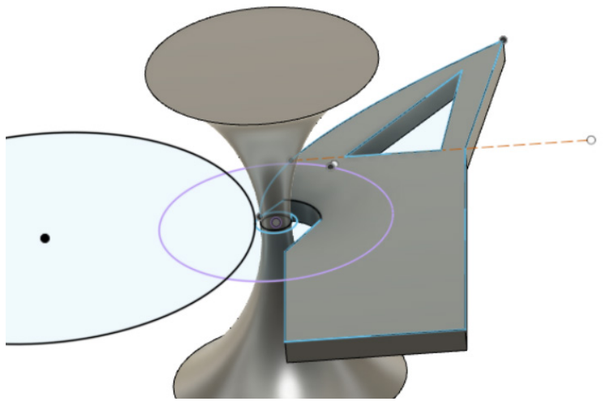
I am not sure whether this will work so I will experiment with it. I think that if I have an Infra-Red light under the robot and an IR sensor then I will be able to detect whether the robot is on top of the white court markings. This is because these will reflect the IR rays however the other surfaces won't so the IR sensor will not receive any light back.

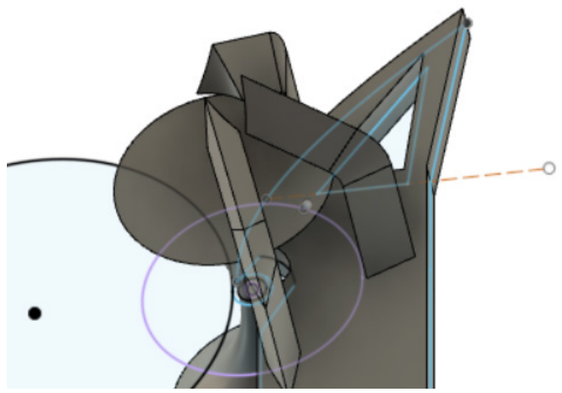
This sensor would be useful for keeping the robot off the tennis court when it is patrolling around and on it when it is in practice mode.

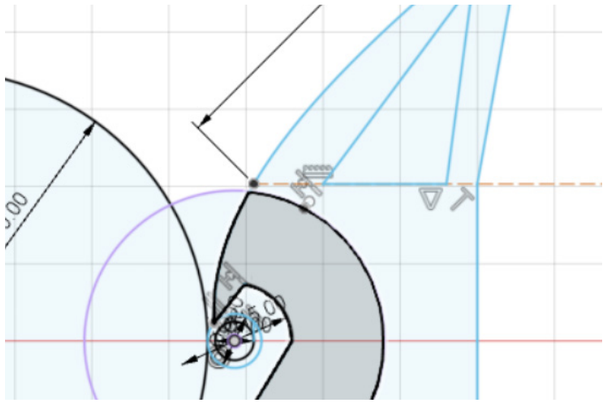
Compass

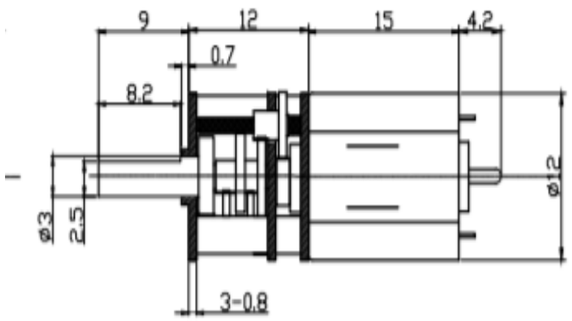
In practice mode, it needs to be able to go straight and then turn 90 degrees. A compass would really help it do this.

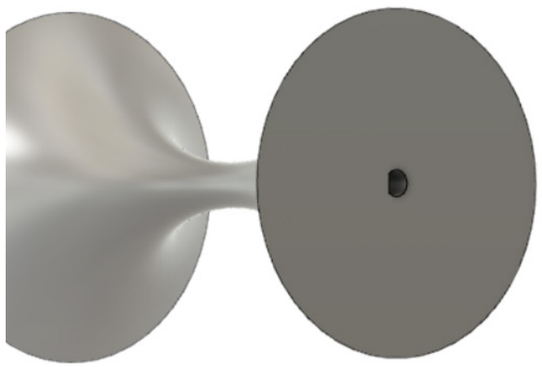


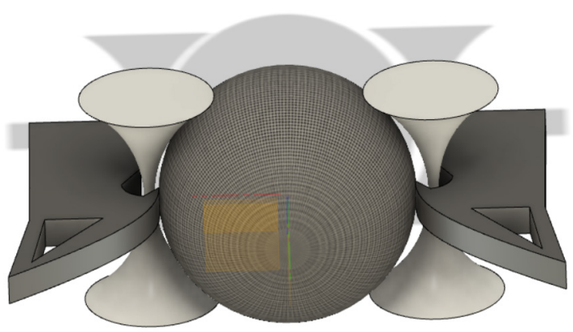


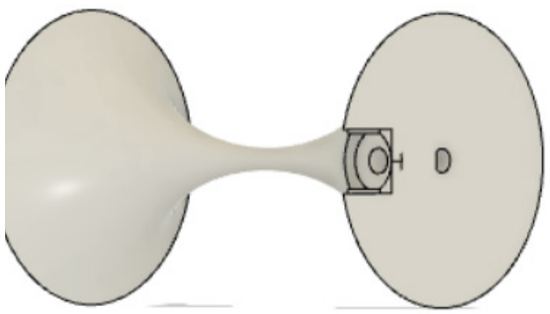






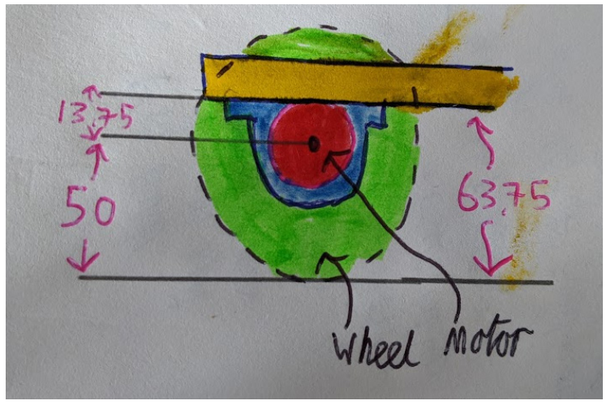


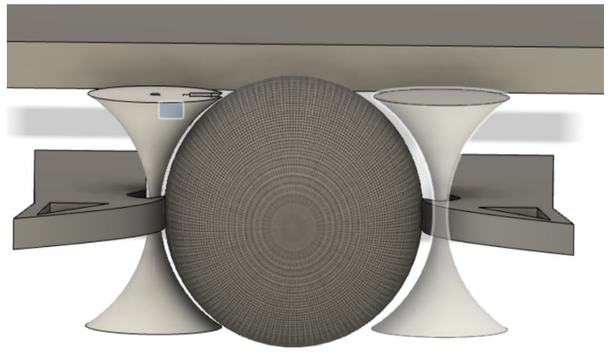


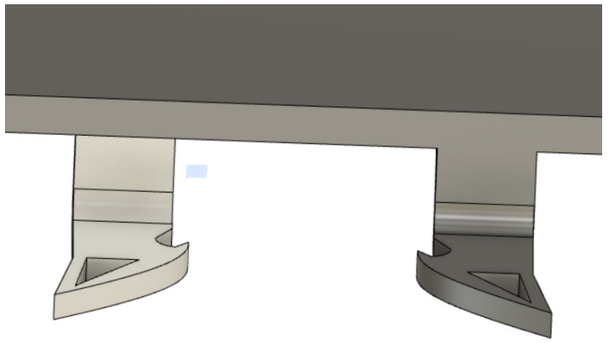


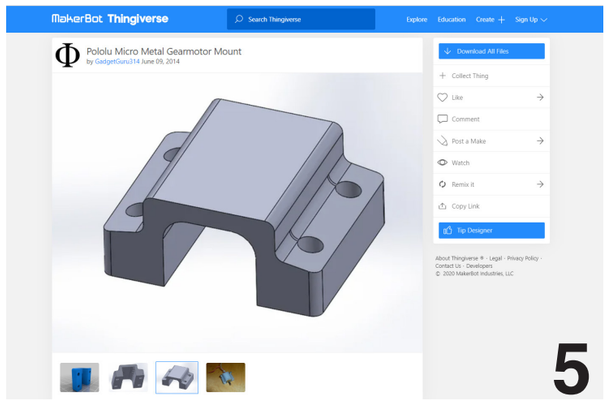
Making the prototype

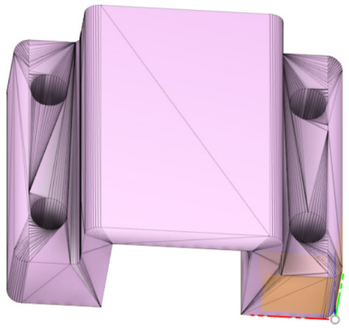
I decided it would be best to 3d print all the complicated parts of the prototype as this was the quickest way to see if my idea would work. I designed all the files in fusion 360. First I designed the rollers that would propel the ball. I did this by sketching 2 circles the size of a tennis ball and taking half of what was in the middle to rotate around 360 degrees. I also needed 2 ramp like things to position the ball before it was propelled. I designed these around the centre of the spinners so the ball would be alligned exactly. My first idea for how to hold the motor that would power the spinners is shown in the picture to the left. However, I later came up with a new idea to attach the motor to the spinners. After I looked up the dimensions and cut out and exact D-shape hole for it to fit in, I then cut out another hole that a screw could fit in to hold the spinners on. I worked out how high the piece of wood I was going to use would be as the chassis would be by adding together half the height of the motor to the wheel radius. I added this to the main file in fusion 360 before attaching the ball positioners to it. I then cut holes in them so I could bolt them to the wood. I decided a better way of attaching the motors would be to hold them up from the top of the wood. I found a motor clamp for the motor on the online website ‘thingi-verse’.

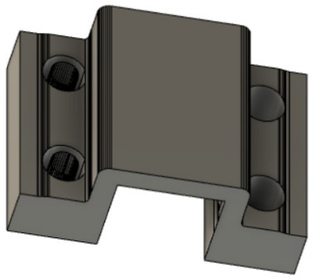


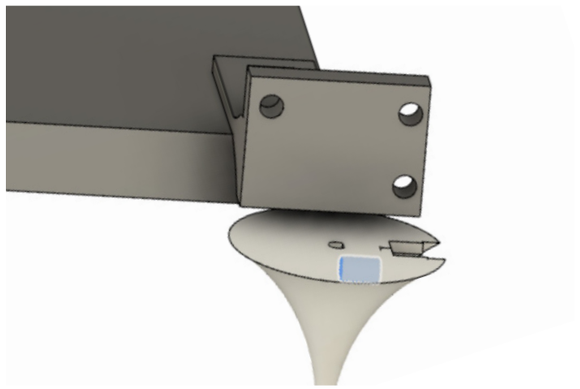


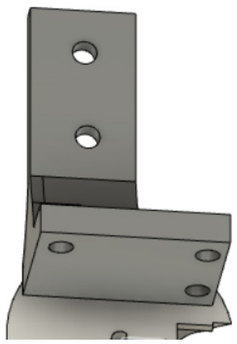


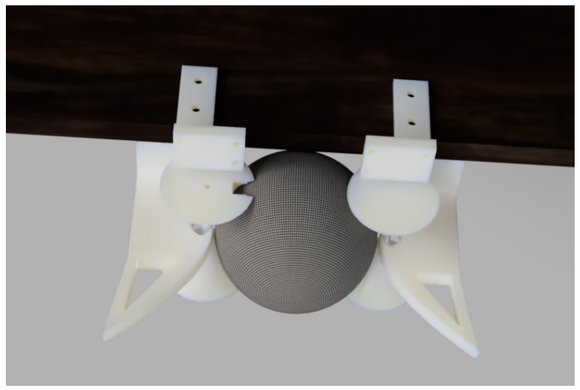


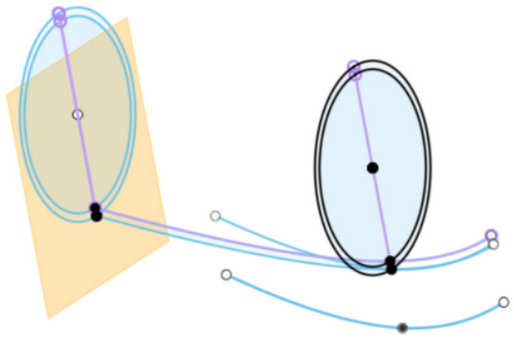


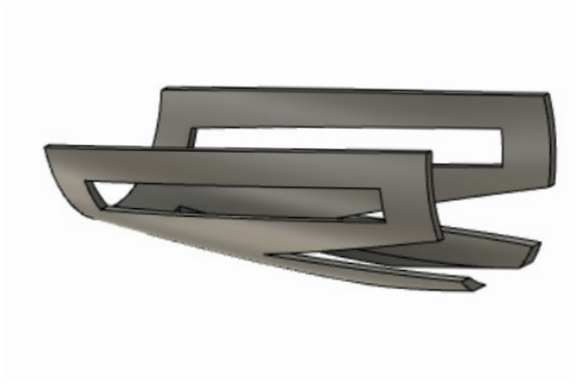


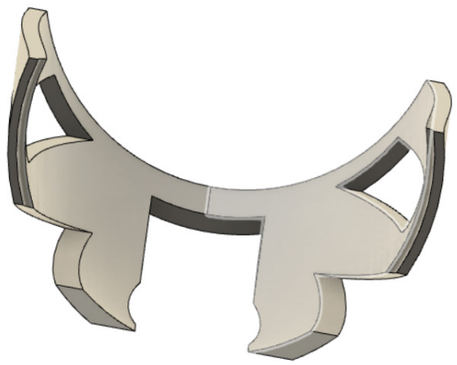


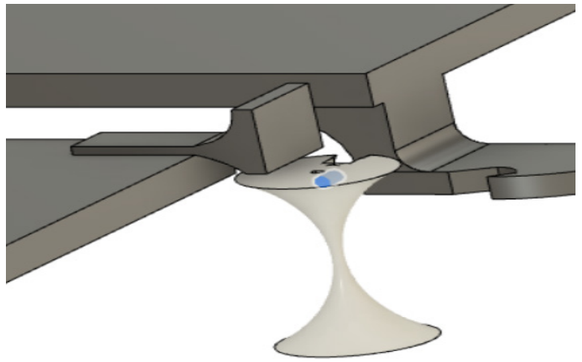


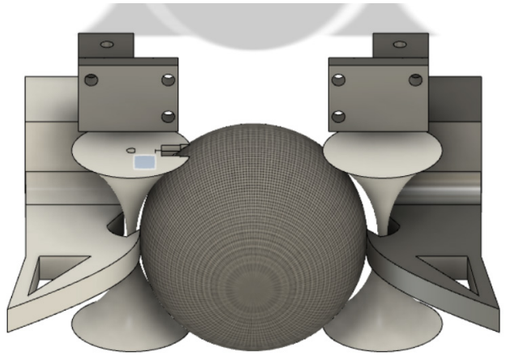




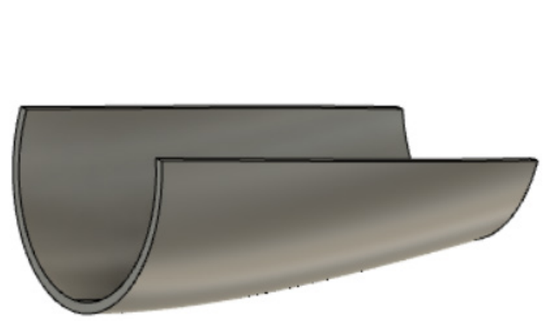


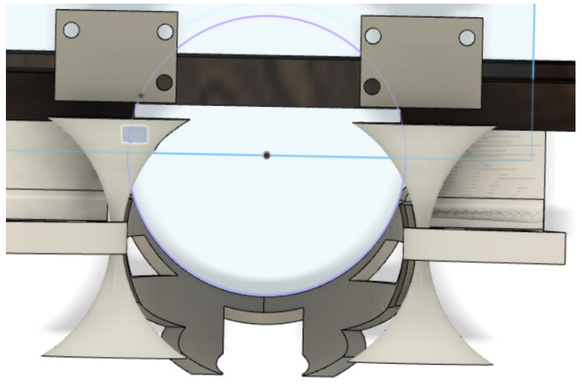


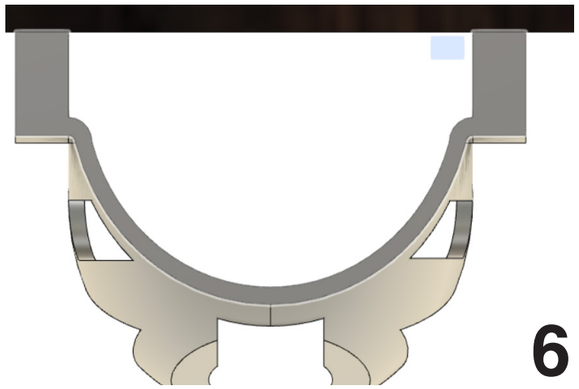


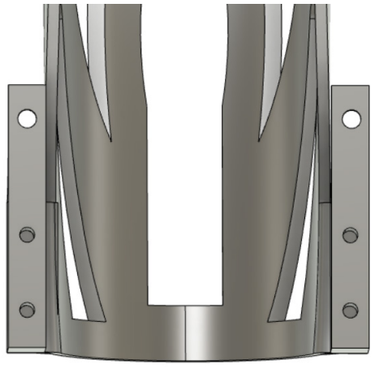


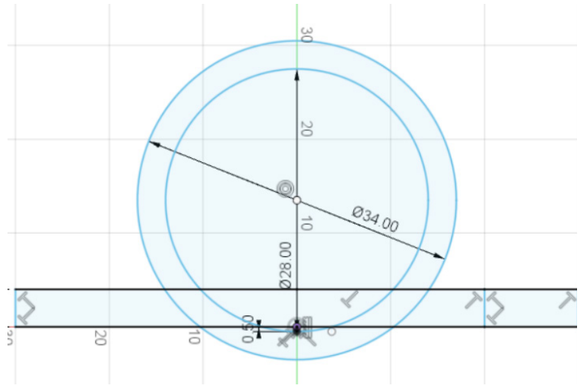
I then imported it to fusion 360 and copied the mesh exactly. I cut out 4 holes in each corner. I made the 4 holes in the same position of the arm that would hold the motors up. I mirrored everything I had made so far so the other side would be identical. Next, I designed the shoot that the tennis ball would be propelled up to enter the tray. I did this using the loft function to join 2 circles (one was 2 cm's higher than the other). I then cut off the top before cutting various other non-essential parts to make it lighter. I then made the front edge curved and altered the bottom edge so the bottom of the shoot could be as close as possible withought touching the ground. I imported it into the main file and made attachments on it to the wood.

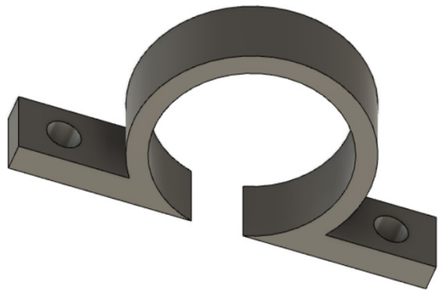




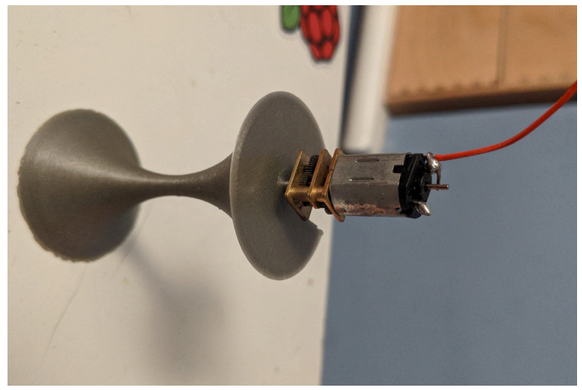


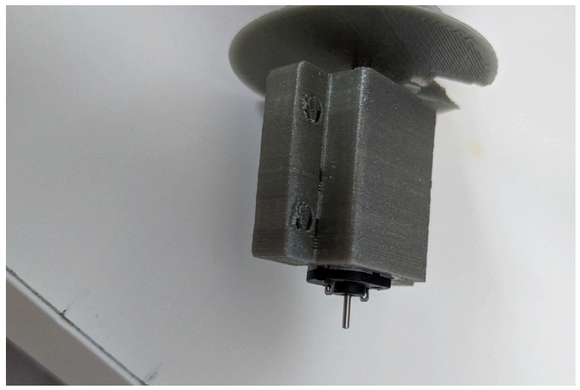


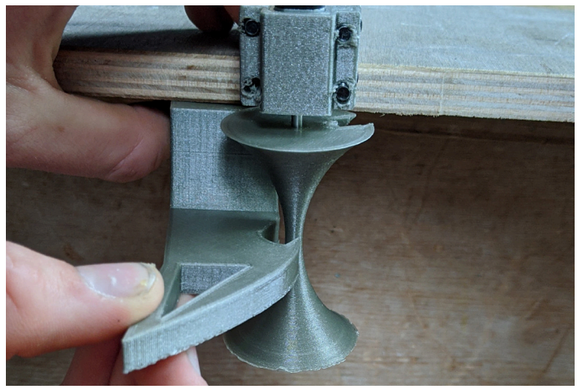


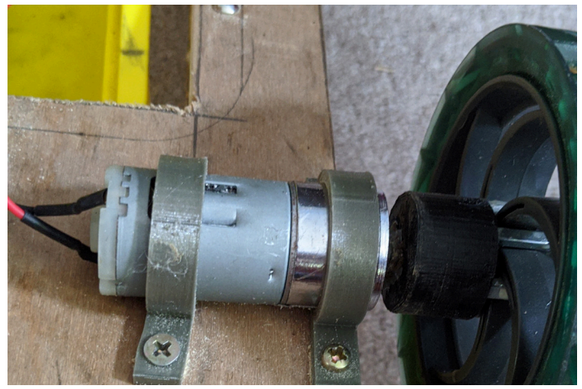


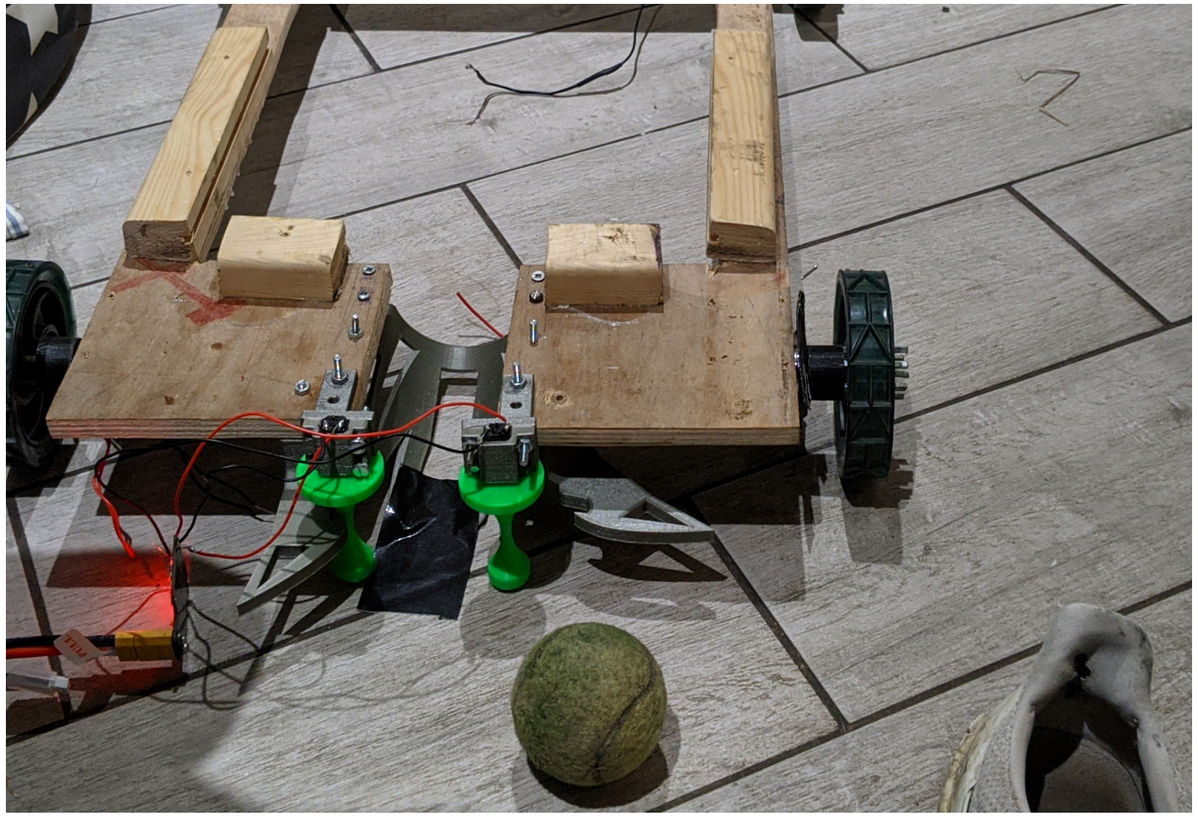
I cut holes in the top of the shoot so I could bolt it to the wood. I then designed a mount for the main motors that would move the whole robot. After that I used a jigsaw to cut out the piece of wood. I cut it out with a hole in the middle for a tray to slide into. I then printed all the parts and attached them to the wooden chassis. I also fitted tray supports so that the tray could be sloped as necessary.

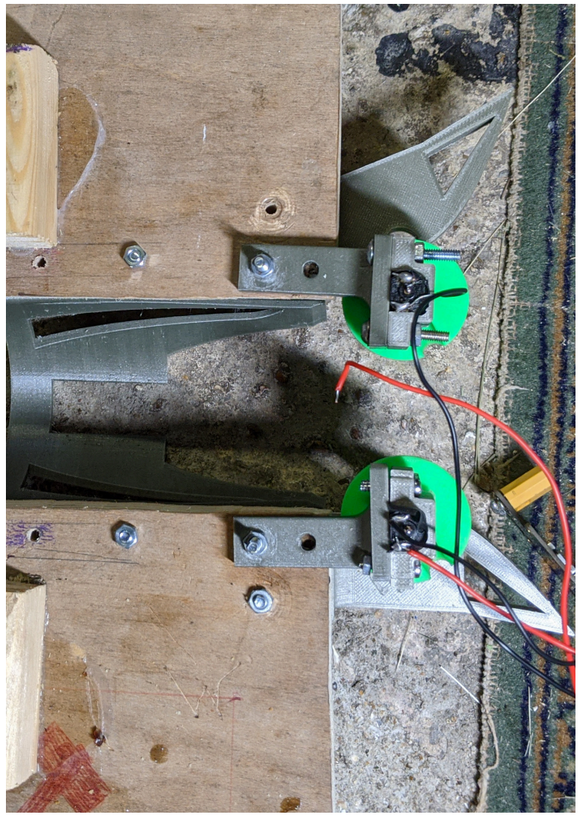




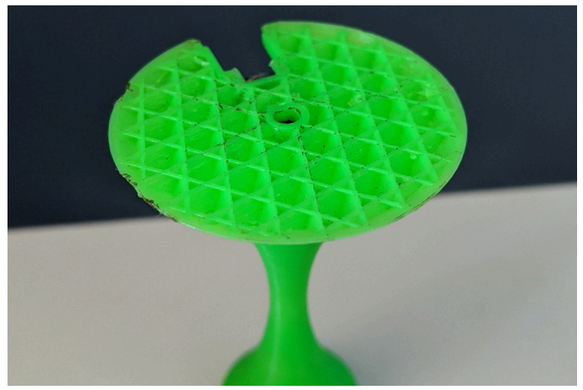


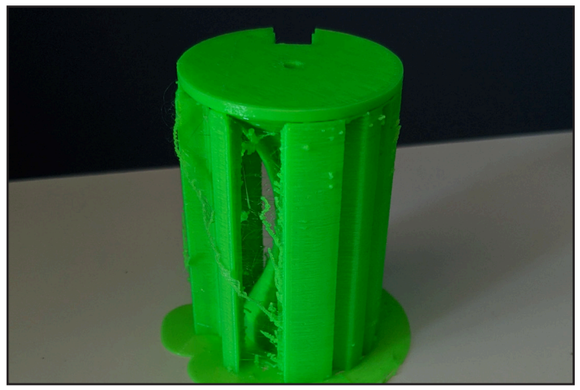


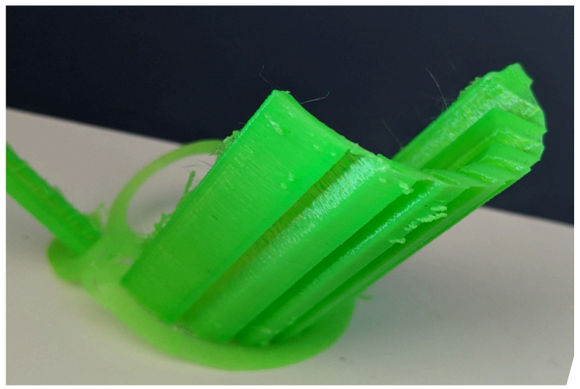


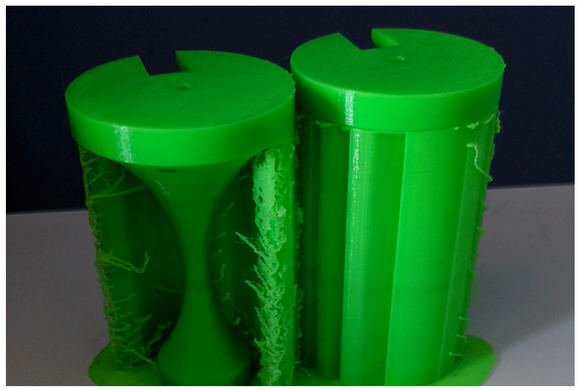


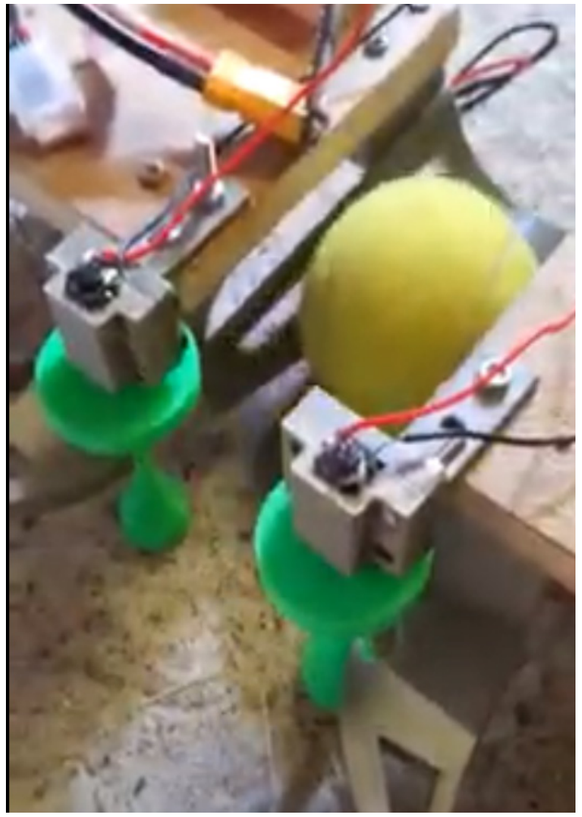
I decided to test the tennis ball propelling system. To do this I solder connected the motors to a power management system designed for drones. It takes in power from a 4S lipo battery and converts it to 5V or 12V.
At first the ball just got stuck and the motors stalled but I adjusted the width between the two propelling spinners by unscrewing the motor holders and changing the angle. However, it still only went about half way up. I had a theory that this was because the spinners were too high up, so I printed some more which should have been lower down as I extended the top bit. However, this made them too top heavy to print and they toppled over. To solve this I printed them using support which I could then tear off. They then printed successfully but the ball still didn't go all the way up the ramp.



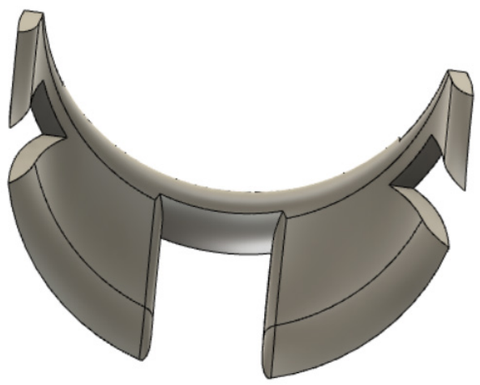


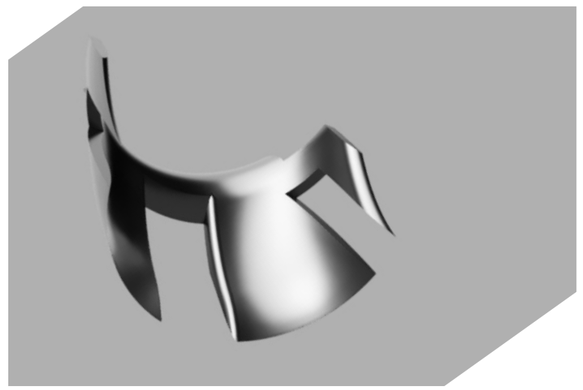


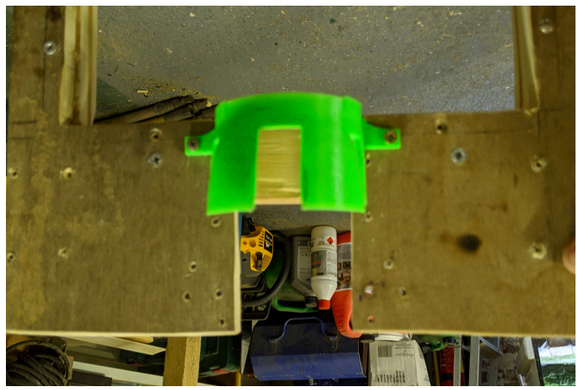


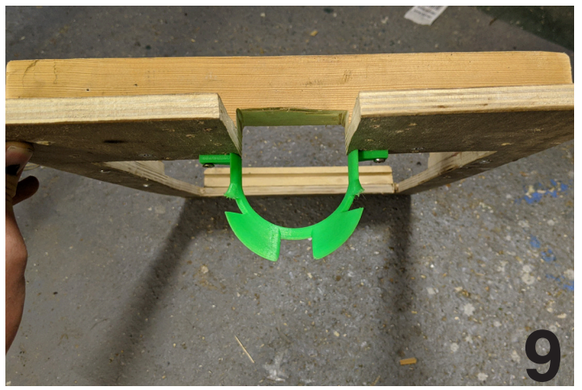


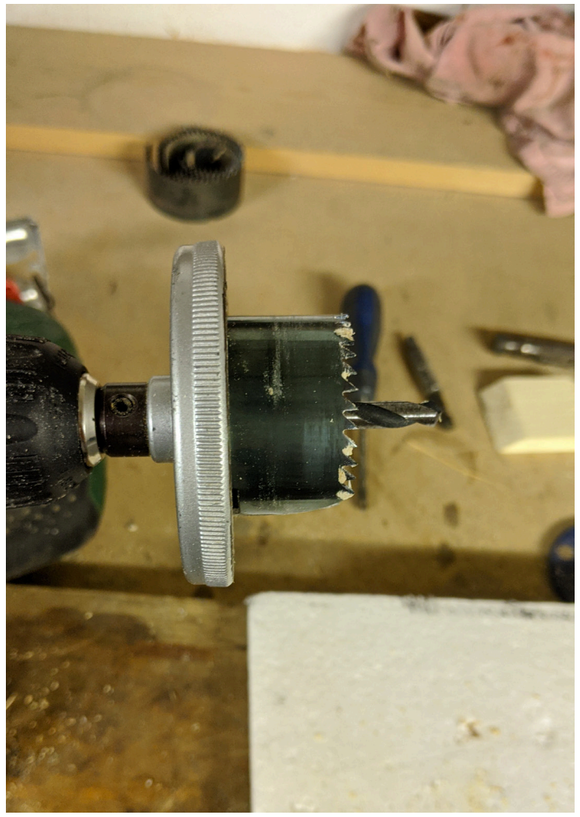
I then tried to give the spinners more grip by adding masking tape. This did not work so I tried using a microflame to give it more texture. This melted it so I tried a soldering iron. This worked a bit better and the ball went about 3/4 of the way up. I then thought that the issue might have been caused by the fact that the bit of wood was not stable and wobbled about a bit. To solve this I added a beam of wood to go across the gap. I then took away the 2 bits of wood that were there before using a chisel. I then used an angle grider to smooth the wood before attaching the new peice of wood with some screws; and using the tray to mark out the ledge I would need to cut out if it. I did the cut out using a table saw before adding some wood glue and screwing it back on. I thought that one of the other problems could be that the shoot was too big so I designed and printed a new one.

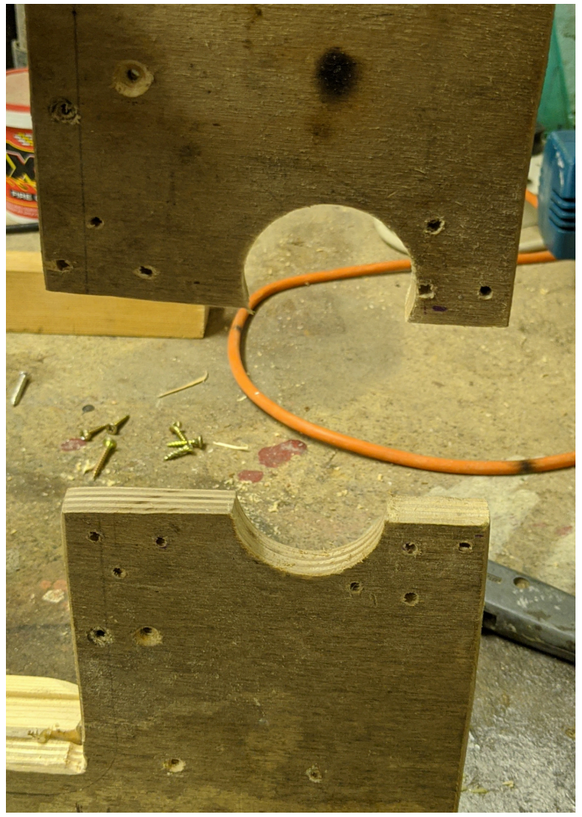




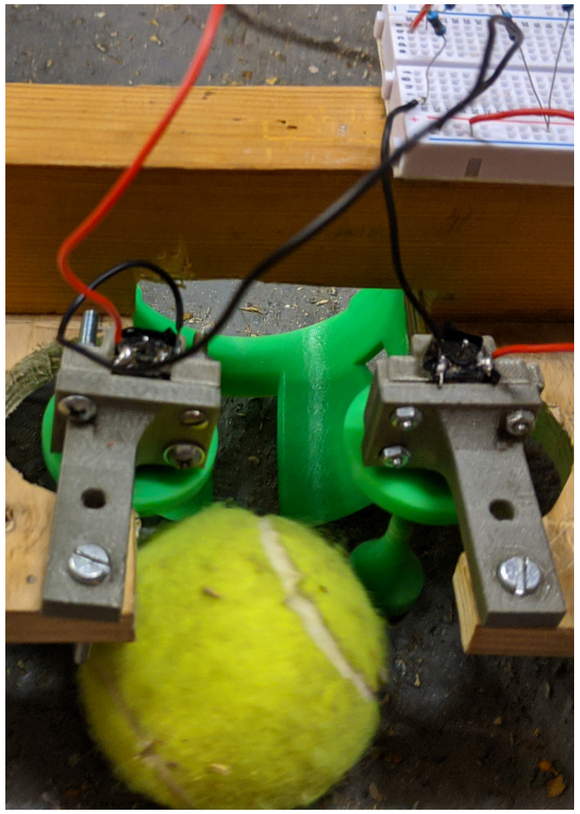


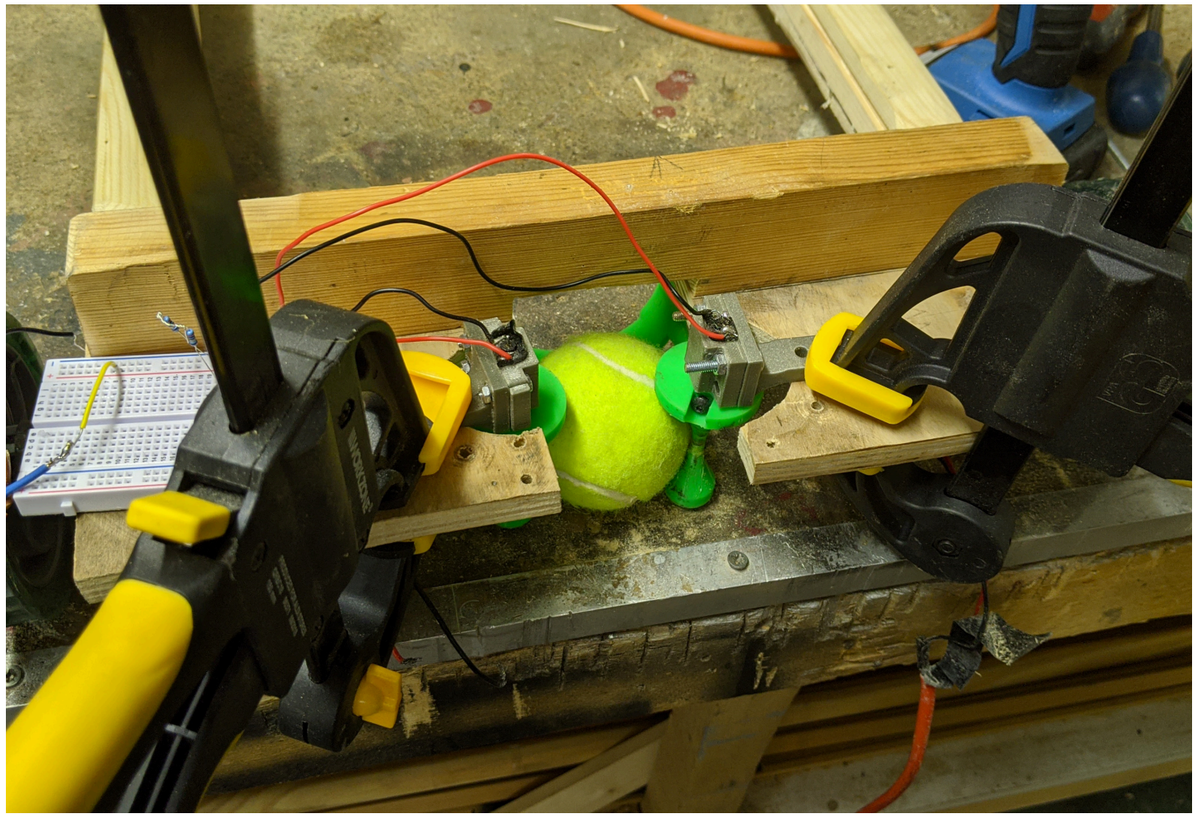


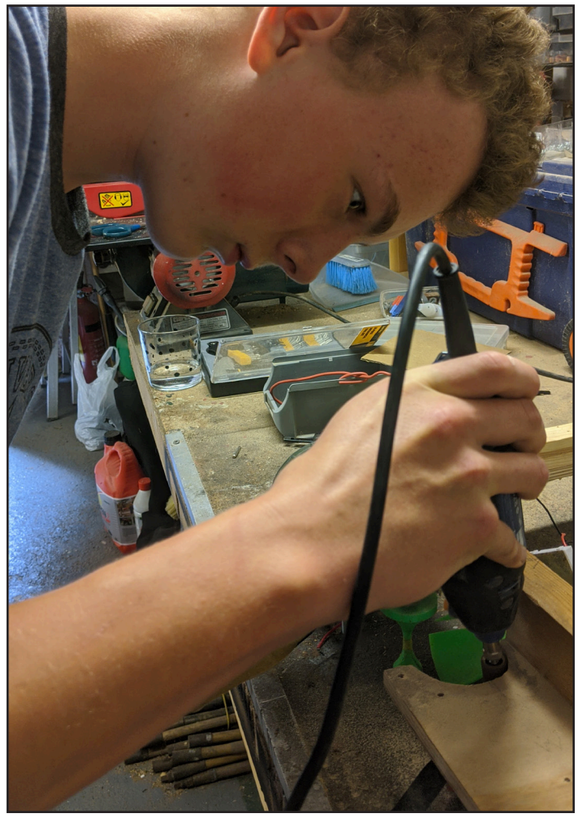


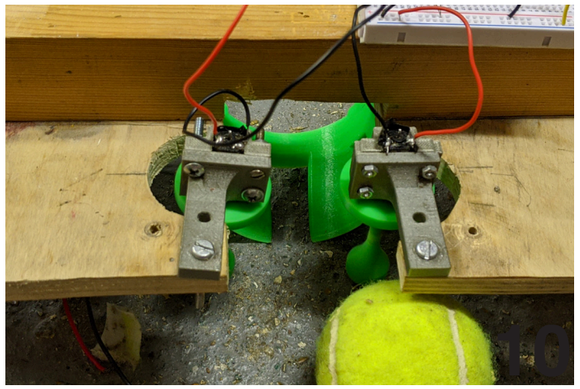


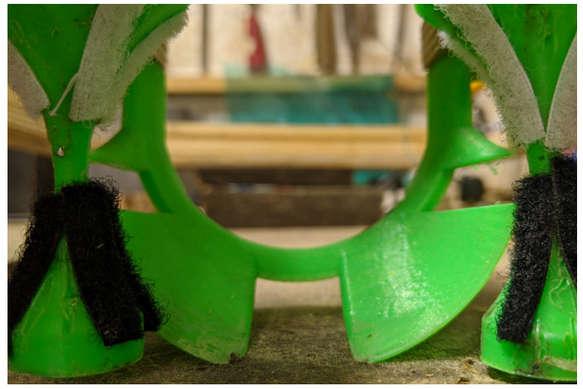
I then used a drill to cut out a circle in the wood where the spinners would spin (closer to the shoot this time so the ball has to go less far), before using clamps to attach the motors to see if the would spin. They didn't as the circles were in the wrong place, so I used a Dremel rotary tool attachment to widen them. I then rescrewed the motors in and tested the mechanism. No luck!

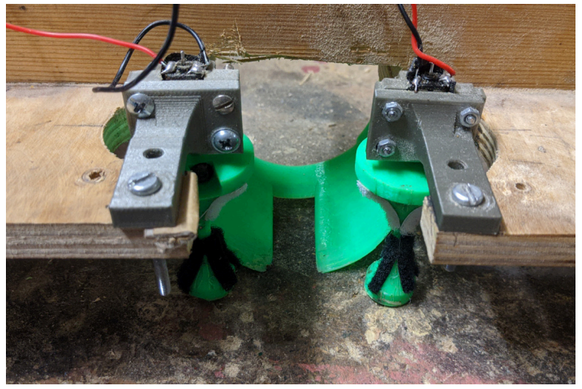




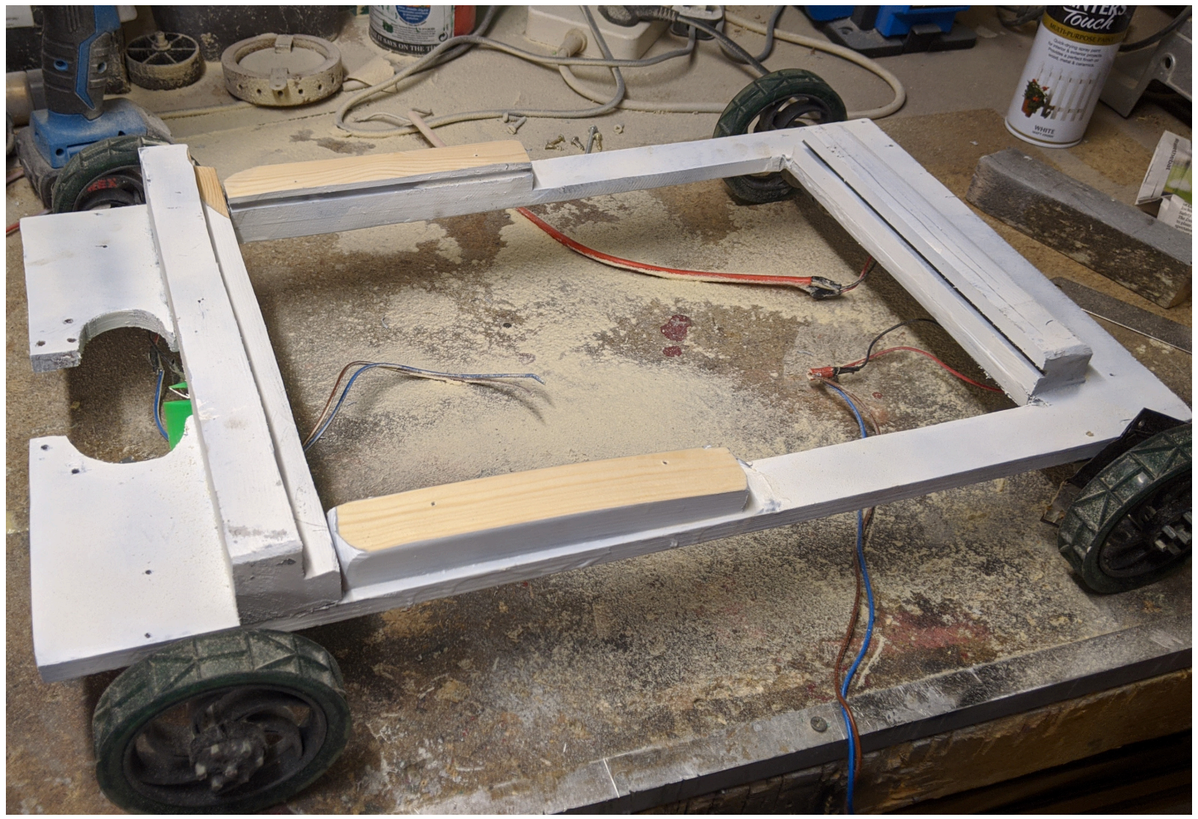


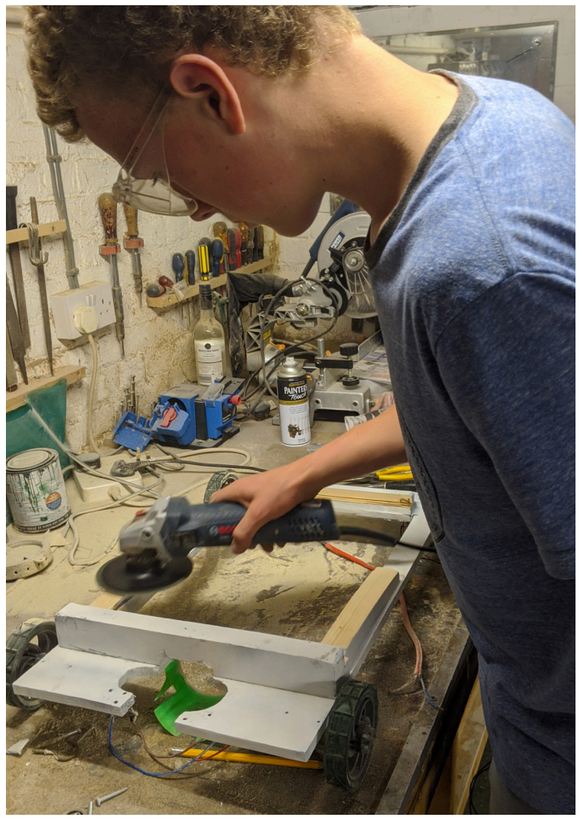


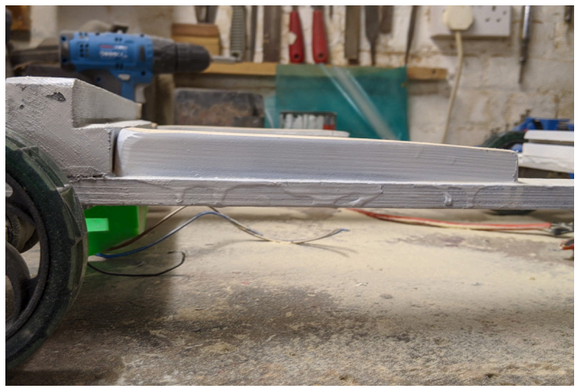


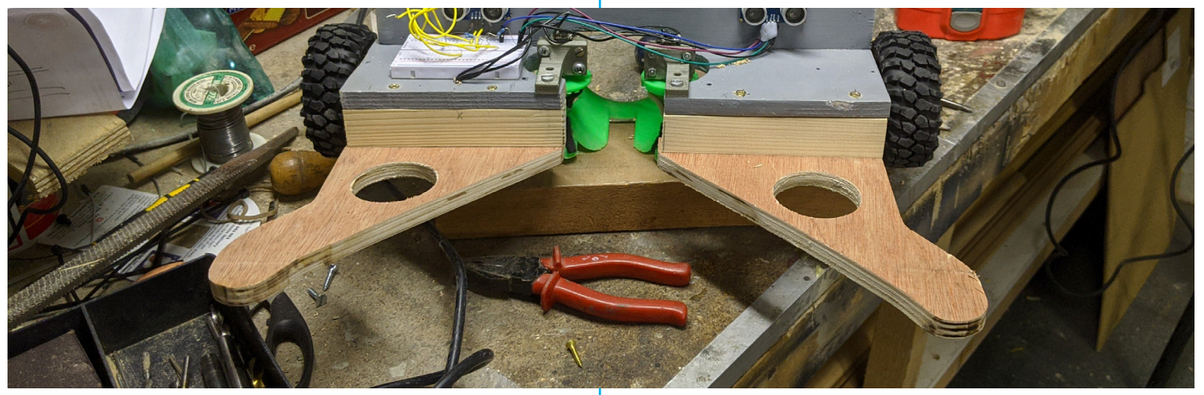


I found some Velcro tape which I added on to the spinners to give more friction. This worked and the ball went up the shoot every time!
To make the robot look better I sprayed some grey paint over it.
I wanted the tray to be at an angle so all the balls would fall to the bottom and not escape back down the shoot or block entry. To do this, I used an angle grinder to slant the bits of wood that held the tray in position. I then used a stanley knife to cut out a hole in the tray.

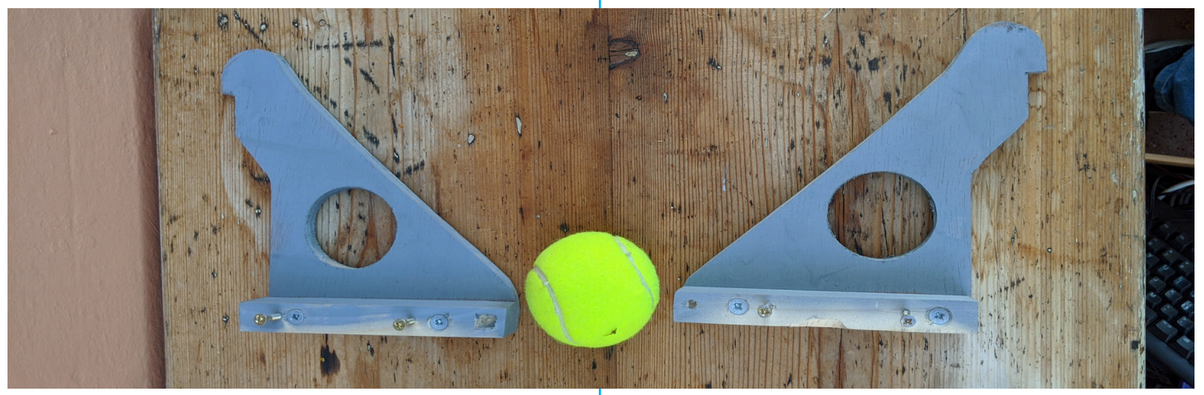


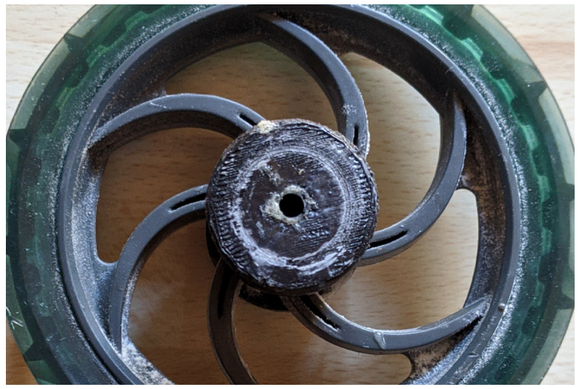


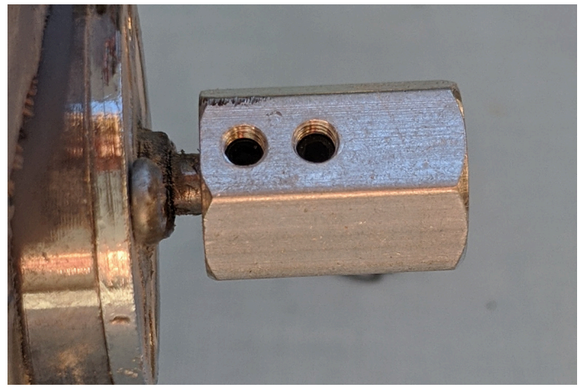


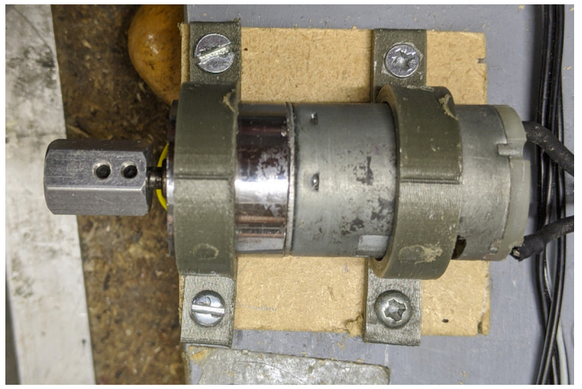


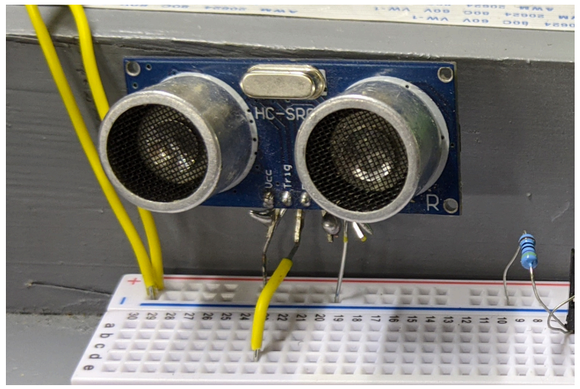
I then reattached the spinners before cutting out 2 guides to position the tennis balls. I used these instead of the previous 3d printed ones as they were a lot bigger so the robot would have to be less accurate to get a ball. While testing I found that the wheels kept on falling off. This was because the wheels I was using were not compatible with the motors and my 3d printed solution to this did’nt really work. I bought some replacement wheels and a proper axle adapter online. However the wheels were smaller so the ramp dragged against the ground. I had to use a piece of wood as a shim under the motors. I used pins to attach 2 distance sensors on the front. To connect these sensors to the raspberry pi, I found an old ribbon cable that came as a spare when I built my 3d printer. I used paper clips to connect it to the breadboard and after sanding a bit of the edge off, it fittered directly to the Pi.

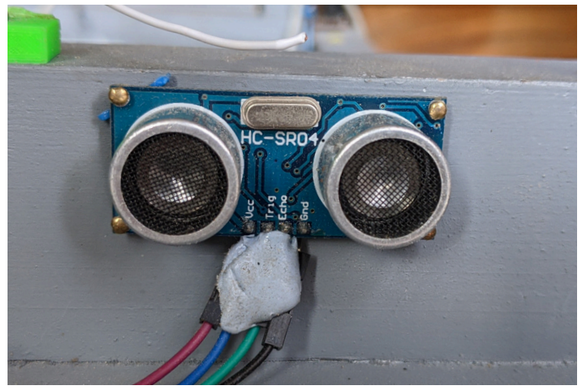


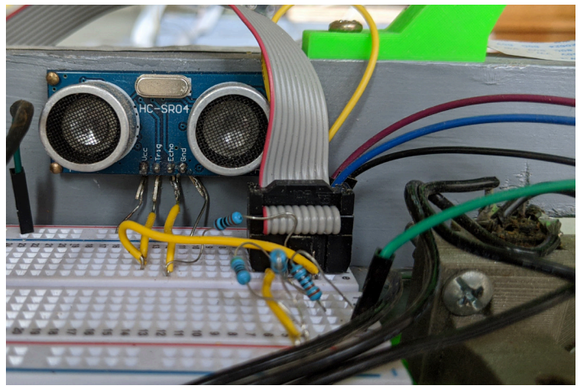


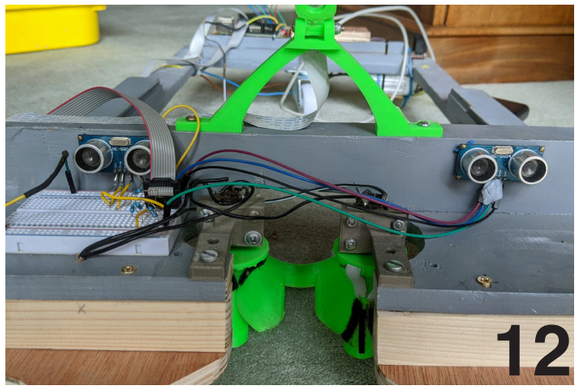


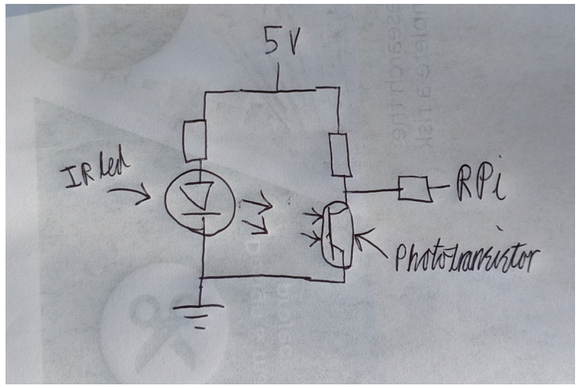


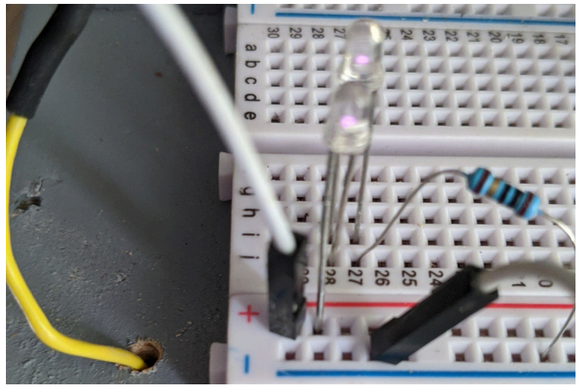




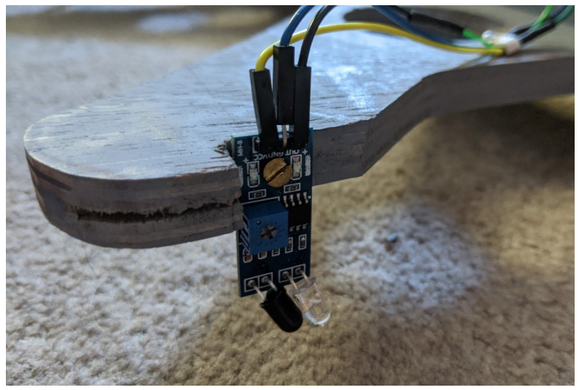


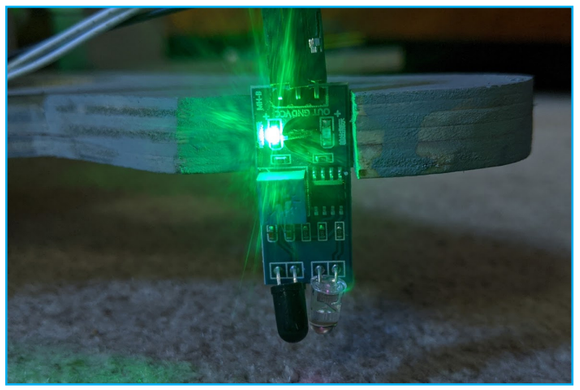


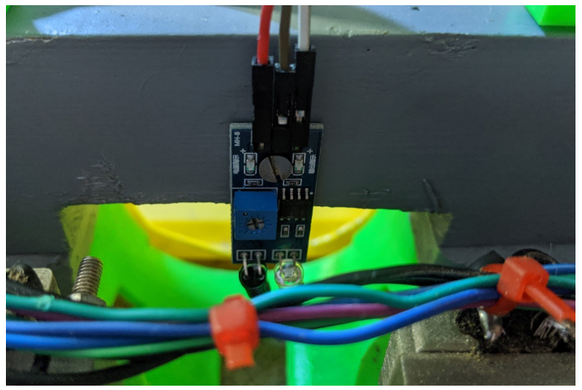


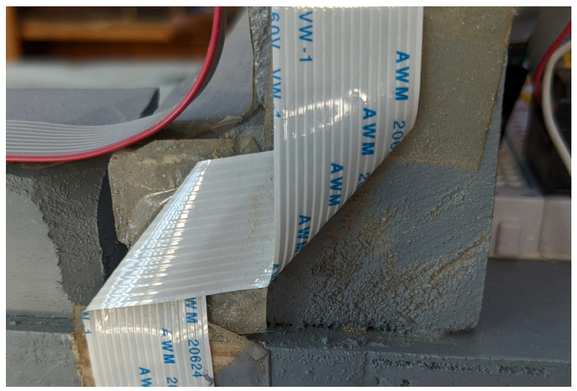


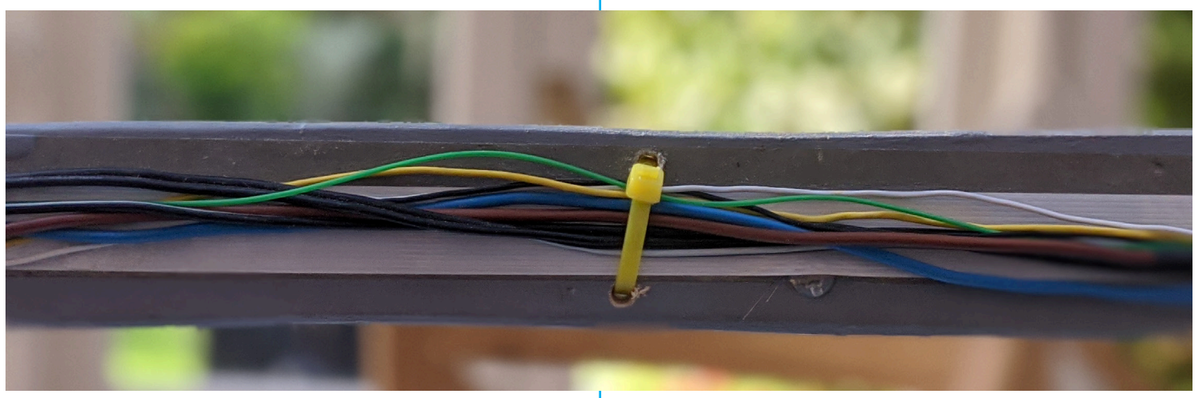
For the robot to stay off the court, it has to know when it passes the white lines. My idea to solve this was to use an IR LED along with a phototransitor. The photo resistor would be in a pull up circuit connected to the rasberry pi. I played around with this a bit but the problem was the raspberry pi's pins are digital so they could only sense an on or a off but nothing in between. This meant my idea would be very difficult to get right. I was trying to find a solution to this on google when I came across line following robots. They use the exact same tech to follow lines you can buy line following modules. I did this and attached 2 of them to the robot. There were now rather a lot of wires so I used double sided sticky tape and zip ties to tidy up the wiring between the front and the back. The robot was now going quite fast so I replaced the smaller caster wheel with a bigger one.

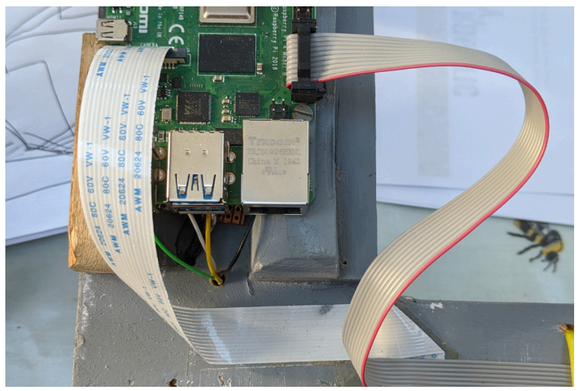


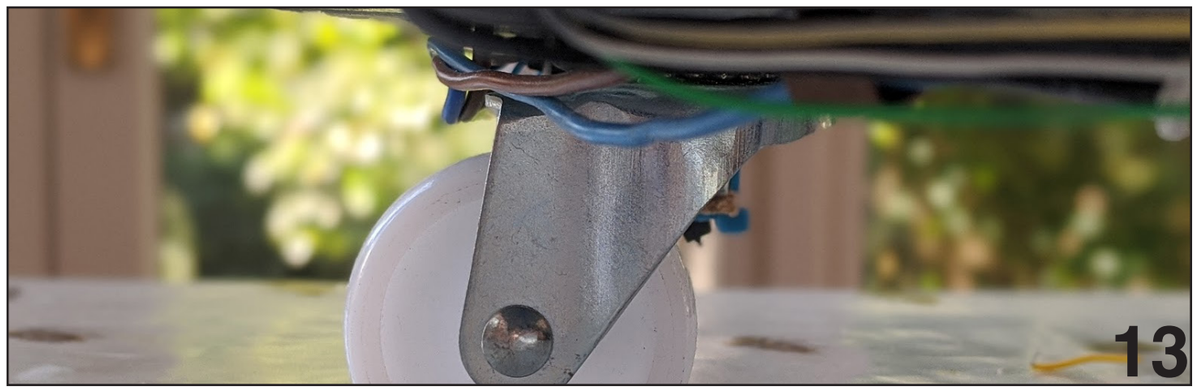










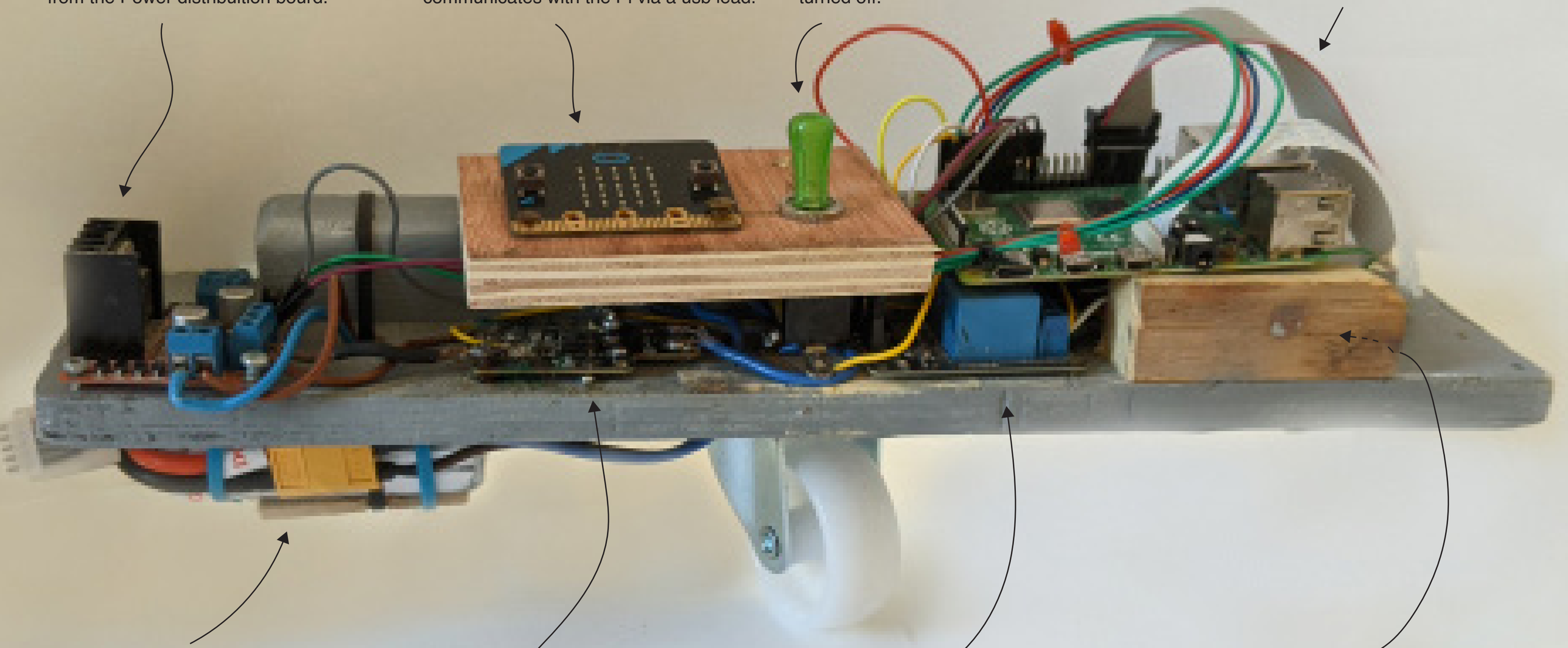


H-Bridge
The H-bridge controls the motors. It takes signals from the rasberry pi GPIO pins to make the Left/Right motor go Forwards Backwards or Off. It gets its power directly from the Power distribuition bourd.

Micro-Bit
The microbit acts as a way for the user to switch between modes and see what the robot is doign via its 25 LEDS and 2 buttons. The Pi also uses it as a compass. It communicates with the Pi via a usb lead.

Switch
This 12V switch allows the user to turn everything off to it is safe to store. The battery goes through it before it gets to the power bourd to ensure everything is turned off.

Raspberry Pi
The raspberry pi is the control centre of the robot and controls everything that happens on it via its GPIO pins which are connected to all the components.



Li-Ion Battery
This is a 4S Lithium Ion (Li-Ion) battery that I bought on amazon. When it is fully charged it provides 16.8V. I have used a zip tie to secure it to the bottom of the robot so it does not fall off. The 2 yellow connectors you can see are XT60 connecters. The wires coming off from this go through the 12V switch to the power bourd.

Power Bourd
This is connected directly (apart form the switch) to the battery and manages its safety. Undervolting Li-Ion batteries can damage them. Also if there are any short circuits, this will cut the power off. It provides a 16-14V (depending on how charged the battery is) for the motors, 12V for the spinners and 5V for the raspberry pi.

Relay
This relay turns the spinners on and off as well as the light on the switch. It is controlled by the GPIO pins on the raspberry pi.

Resistors
The spinners take 9V but there was only a 5V or a 12V output on the power bourd. To solve this I used two resistors to bring the voltage down. I originally had these on a breadbourd but they kept on coming loose so once I knew which ones to use, I soldered them onto some veribourd. I used two instead of one so any heat dissipated would be split between two.

Distance sensors

IR Sensors

Motors for the spinners

Li-Ion 4s Battery

Relay

12V

Camera

Switch

16V (Depending on how charged the battery is)

5V

Power distribution bourd

GPIO

Micro-bit

Raspberry Pi

H-Bridge

USB

Motors

Thinking about the software

Language

I decided to use python as I already know it well. There is also a python version of opencv which is the library I may use to detect tennis balls. Moreover, you can easily use python to control the GPIO pins on the raspberry pi which I will then use to control the motors and the sensors on the robot.

OpenCV or Teachable machine

I will need to use computer vision to detect the tennis balls.

To do this I could either use OpenCV or Google's teachable machine. If I was to use OpenCV to detect the tennis balls, I would first detect all the circles in the image. OpenCV has a built-in function that will do this. From those circles, I could then see which are the right colour(yellow). I would do this by sampling the middle pixels and checking their colour. If the colour is in a predefined range, it would be classed as a tennis ball. OpenCV could then tell the robot, where the middle pixel is in the image, so the robot knows if it needs to turn right or not. Edit: I originally set out to do this but when experimenting with the fastest method, I found that it was a lot faster to just mask all the non-'tennis ball yellow' colours and then anything left would be a tennis ball. I then looked for the biggest 'blob' and then moved towards it

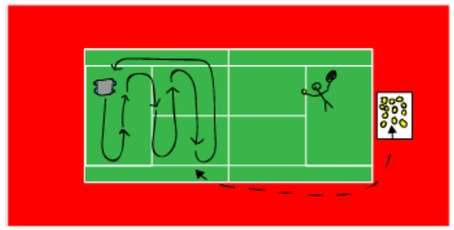
If I was to use Teachable machine, all I would need to do is give the machine lots of photos of tennis balls and lots of photos of no tennis balls, and it would do the rest and the website would give me some code that I could use. The problem with this would be that the robot wouldn't now know where the tennis ball actually is relative to it. Also and probably more importantly, this would take too long as the raspberry pi can't processing things as fast as a normal computer.

Different modes

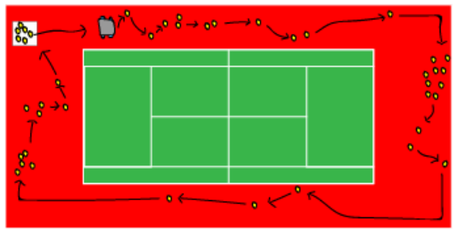
Depending on what the tennis players are doing the robot will have to act differently.

Here are the modes I have considered:

- Practicing
    - 1 player is just continually serving hundreds of shots onto an empty court on the other side.
    - My robot would be able to allow them to do this infinitely as it would patrol the other side and periodically bring tennis balls back to the player.



- Playing club level'
    - The robot would patrol around the edge of the courts by the wall or net. It would never go onto the playing court incase it disrupted the players.
    - It would know to turn at the edges because I could give it an ultrasonic distance sensor so it would be able to sense the court netting coming up. It would then turn exactly 90 degrees (on board compass).
    - It wouldn't go onto the court because of the IR sensor.
    - At one corner there would be a tennis ball bin where it will park and wait until its tennis balls have been collected. It would know its there because it would have come to the 4th corner.



Thinking about the software

- Playing professionally
    - I decided not to worry about professional games as it would not be required:
        - They do not use as many balls
        - They have ball boys already

You could set the mode via the micro bit attached to the robot.

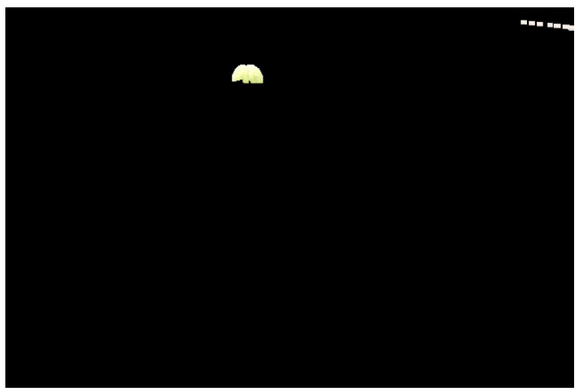
```
#Import libraries
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
from gpiozero import LED,DistanceSensor,LineSensor,Motor, Button
import serial
import math

#Constatnts
ANGLE_MULTIPLIER = 10
CRASH_DISTANCE = 0
FOWARD_TIME = 2

IMAGE_WIDTH = 640


#Tries to connect to the microbit
try:
    PORT = "/dev/ttyACM0"
    BAUD = 115200
    s = serial.Serial(PORT)
    s.baudrate = BAUD
    s.parity   = serial.PARITY_NONE
    s.databits = serial.EIGHTBITS
    s.stopbits = serial.STOPBITS_ONE
    print("Microbit detected")
except:
    print("No microbit detected")
    exit()
#Filter values to filter 'tennis ball yellow' for image processing
VALUE = [29,86,6,64,255,255]
print("Filter:",VALUE)
#Lower and upper yellow filter using values above - To be used in BGR part of the
#filtering
BGR_LOWER_YELLOW = np.array([VALUE[0], VALUE[1], VALUE[2]], dtype=np.uint8)
BGR_UPPER_YELLOW = np.array([VALUE[3],VALUE[4],VALUE[5]], dtype=np.uint8)
VALUE = [20,1,1,60,220,255]
#These values will be used in the BGR stage
HSV_LOWER_YELLOW1 = np.array([VALUE[0], VALUE[1], VALUE[2]], dtype=np.uint8)
HSV_UPPER_YELLOW1 = np.array([VALUE[3],VALUE[4],VALUE[5]], dtype=np.uint8)

#Set up camera and give it time to warm up
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640, 480))
time.sleep(1)

#Setup distance sensors
sensor1 = DistanceSensor(19, 6)
sensor2 = DistanceSensor(26, 13)

#Set up IR sensors
IR1 = Button(2)
IR2 = Button(3)
IR3 = Button(22)

#Set up motors
lmotor = Motor(27,17)
rmotor = Motor(23,24)

#GPIO pins to control the relay
spinners = LED(14)
switch = LED(15)

#Turning the spinners off but the light on to indicate the program has started(If you say on, it turns it off and vice versa)
switch.off()
spinners.on()

#Compass - returns a bearing (0 to 360) and stops if it cant access the compass
def compass():
    attempts = 0
    while True:
        attemps += 1
        try:
            data = s.readline().decode('UTF-8')
            data_s = data.rstrip().split(' ')
            x, y, z, a, b, t = data_s
            return t

        except:
            pass
        print("Stuck in a loop :)")
        if attempts > 10:
            print("Compass not working")
            exit()

#Functions for movement
def stop():
    lmotor.stop()
    rmotor.stop()
def simpleforward(distance,sped):
    print("forward",distance)
    spinners.off()
    lmotor.forward(speed = sped)
    rmotor.forward(speed = sped)
    time.sleep(distance)
    stop()

#This function allows the robot to go forward without crashing into anything.
#It goes a set amount.
def forward(Grad_Stop = 0):
    global FOWARD_TIME
    global CRASH_DISTANCE
    lmotor.forward(speed = 0.5)
    rmotor.forward(speed = 0.5)
    spinners.off()
    old = time.time()
    while time.time() <= FOWARD_TIME:
        #Checks to see if any of the distance sensors sense something
        if sensor1.distance < CRASH_DISTANCE or sensor2.distance < CRASH_DISTANCE:
            stop()
            #If both sensors sense something:
            if sensor1.distance < 1 and sensor2.distance < 1:
                #This works out the angle the robot is to the wall
                if sensor1.distance > sensor2.dinstance:
                    o = sensor1.distance - sensor2.distance
                    a = 0.14#Distance between the distance sensors
                    angle = math.degrees(math.atan(o/a))
                    #If the angle is less than 10, it has probably reached a corner
                    if angle < 10:
                        corner()
                    else:
                        turn(direction)
                if sensor1.distance < sensor2.dinstance:
                    o = sensor2.distance - sensor1.distance
                    a = 0.14#Distance between the distance sensors
```

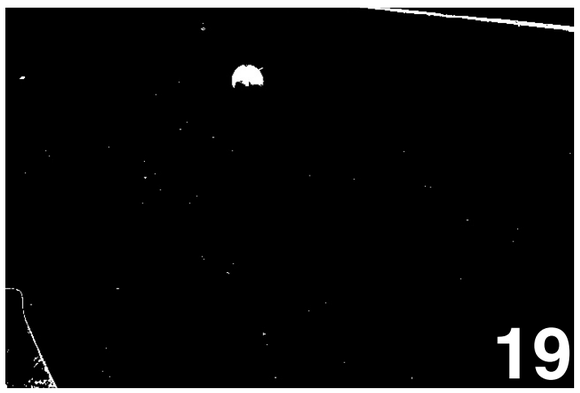
```
            angle = math.degrees(math.atan(o/a))
            #If the angle is less than 10, it has probably reached a corner
            if angle < 10:
                corner()
            else:
                turn(direction)
        else:
            turn(direction)
    #Checks IR sensors
    elif IR1.is_pressed or IR2.is_pressed:
        print("IR sensors have detected something")
        stop()
        #Doubles checks the sensors
        time.sleep(0.5)
        #If the left IR sensor detects something, go right if right then vice versa
        if IR1.is_pressed:
            right(30)
            simpleforward(0.5,0.7)
            turn(direction)
        elif IR2.is_pressed:
            left(30)
            simpleforward(0.5,0.7)
            turn(direction)
        else:
            lmotor.forward(speed = 0.5)
            rmotor.forward(speed = 0.5)
            old += 0.5
#If the option gradual stop variable is not used it defaults to 0
if Grad_Stop == 0:
    stop()
#If it is used
else:
    #Gradually decrease the speed
    #This is so balls dont bounce away
    i = 0.5
    while i > 0:
        lmotor.forward(speed = i)
        rmotor.forward(speed = i)
        i = i-0.1
        time.sleep(0.05)
    stop()
def backward(distance):
    print("backward",distance)
    lmotor.backward(speed = 1)
    rmotor.backward(speed = 1)
    time.sleep(distance)
    stop()
def right(forward_distance,turn_time):
    spinners.off()
    simpleforward(ford,0.6)
    rmotor.forward(speed = 0.7)
    time.sleep(turnt)
    stop()
def left(forward_distance,turn_time):
    spinners.off()
    simpleforward(ford,0.6)
    lmotor.forward(speed = 0.7)
    time.sleep(turnt)
    stop()
#This function turns to a particular bearing using a compass
def turn(angle):
    x = angle - compass()
    x = x%360
    angle = angle % 360
    if x > 180:
        lmotor.forward(speed = 1)
        rmotor.backward(speed = 1)
        while compass() > angle:
            time.sleep(0.01)
        stop()
    else:
        lmotor.backward(speed = 1)
        rmotor.forward(speed = 1)
        while compass() < angle:
            time.sleep(0.01)
        stop()
#Going round a corner
def corner():
    global direction
    turn(direction + 90)
    direction = compass()

#Sets original direction
direction = compass()

#Main loop
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    #Turns spinners off while doing image processing to save energy
    spinners.on()
    #Grap a picture from the pi cameras feed
    #I originally just got the camera to take a photo but this was slower
    frame = frame.array
    #Masking. I masked the image first in the BGR colourspace then in the HSV as both of
    #them didnt work individually
    mask = cv2.inRange(frame, BGR_LOWER_YELLOW, BGR_UPPER_YELLOW)
    #Erroding away insignifgant pixels
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
    #Creates an image with only the bits that got through the first filter
    #(The mask is in Black and White)
    bitwiseAnd = cv2.bitwise_and(frame, frame,mask=mask)
    #Converts colourspace
    hsv = cv2.cvtColor(bitwiseAnd, cv2.COLOR_BGR2HSV)
    #Masking again but this time in hsv colourspace
    mask1 = cv2.inRange(hsv, HSV_LOWER_YELLOW, HSV_UPPER_YELLOW)
    #Finds the contours of the image so if there are 2 tennis balls, it differentiate
    #between them
    cnts = cv2.findContours(mask1.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]
    #Only goes on if atleast 1 body is detected
    if len(cnts) > 0:
        #Processes the biggest one
        c = max(cnts, key=cv2.contourArea)
        #Finds location of body by finding the info about a circle that could fit in it
        ((x, y), radius) = cv2.minEnclosingCircle(c)
        print("Radius:",radius)
        #If the circle is big enough:
        if(radius > 10):
            print(x,y)
            #Draws a circle onto the image so user can see where it thinks the circle is
            cv2.circle(frame, (int(x), int(y)), 10, (0,0,0))
            if y > 400:
                forward()
            if x > IMAGE_WIDTH+10:
                print("left")
                left(2/radius,((IMAGE_WIDTH/2)-x)/750)
            elif x < IMAGE_WIDTH-10:
                print("Right")
                right(2/radius,(x-(IMAGE_WIDTH/2))/750)
            else:
```

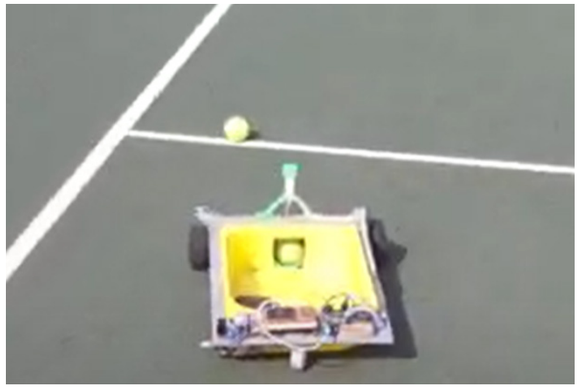
```
            #Goes forward and then waits a second to allow ball to go up shoot before
            #it moves on
            print("forward")
            #this forward will stop slowly as 1 is given
            forward(1)
            time.sleep(1)
    else:
        #If nothing is sensed it tries again a couple of times before
        #looking around and eventually moving forward
        attempt += 1
        if attempt > 7:
            turn(COMPASS)
            forward()
            attempt = 0
        elif attempt > 6:
            turn(compass()+50)
            time.sleep(0.2)
        elif attempt > 5:
            turn(compass()-25)
            time.sleep(0.2)

#Shows frame with circle drawn on it where the tennis ball is
cv2.imshow("f",frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
rawCapture.truncate(0)
```

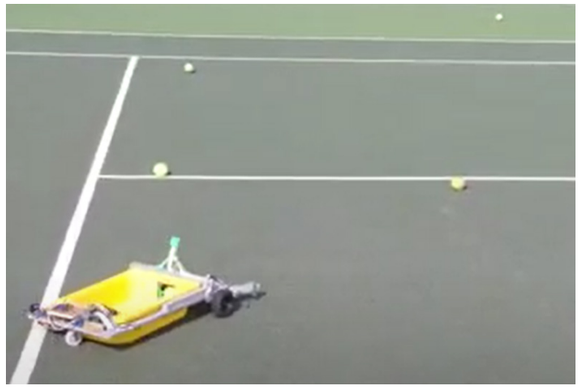
How the image proccessing works

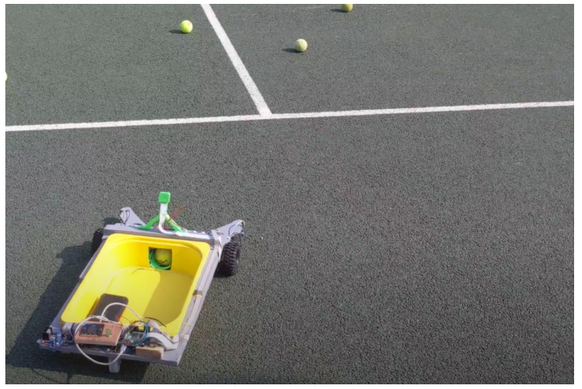
First the scipt grabs a photo from the picameras raw feed. It then filters every pixel in that image by only allowing pixels through whose colours fit within a certain range. The program then erodes the black and white mask created by this using the cv2.erode() and dilate functions. This gets rid of any random dots that have got through. I found that this wasnt enough and by switching colour spaces and repeating the process, I could get more of the background out so I did this and repeated the process in the HSV colourspace. The code then uses the cv2.findcountours() function to create a list of all the different ‘blobs’. It assumes the biggest one is the ball. It might have been more effective to have some machine learning detecting the ball but this is faster and does the job.





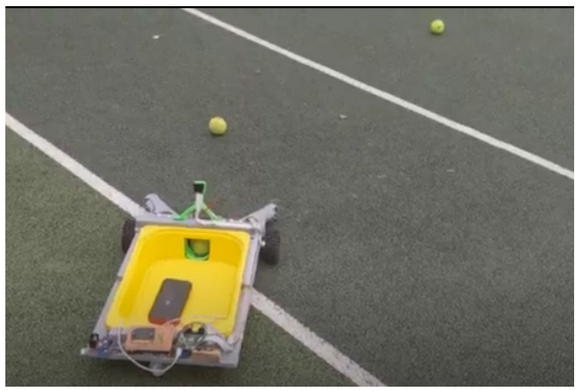


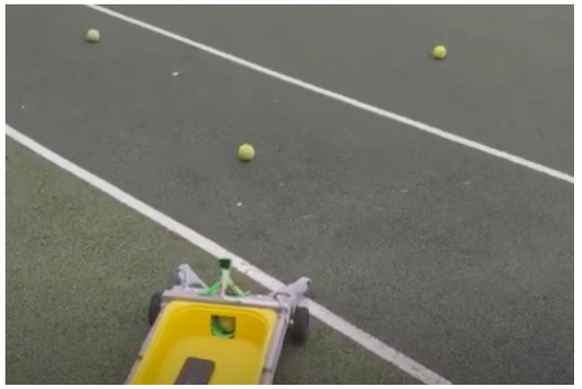


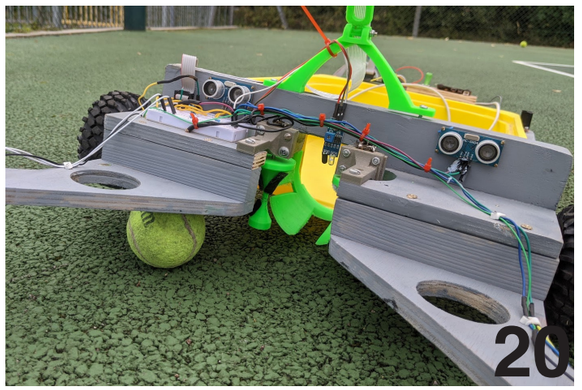


The Robot in action

Testing the robot: After many alterations to the code, I got the robot to work pretty well. If there was a ball within about 3 metres of it, it would collect the ball 9/10 times. The ball didnt always go up the shoot but if it didnt, it would always stay within the spinners and the shoot meaning when the robot collected another tennis ball, the new one would push it up the shoot and replace it. All the sensors worked: The compass, the IR sensors and the ultra-sonic distance sensors. One problem was sometimes, when it went to get a ball, the ball got stuck underneath the guides meaning the whole robot got stuck. One solution to this would be to lower the guides a bit or make them thicker. Another problem was that it quite liked the posts used to hold the floodlights and occasionly repeatably crashed into them as shown in one of the photos at the bottom. In terms of speed, it took about half a minute to get a ball once it had seen it. I had this faster but slowed it down, because when the turning speed was high, it often overturned and lost sight of the ball. I have compiled some cuts of it getting a ball and made a video of them avadible to see here: https://vimeo.com/461877865







Project Diary

22/06/20

- I talked to an actual tennis player and made a brief with all the requirements of my final product
- I researched into previous solutions
- I wrote part of the report about previous solutions
- I started to think about different ways it could capture the tennis ball
- (~2 hours)

23/06/20

- I made a mind map about potential ways to capture tennis balls.
- (~1 hour)

24/06/20

- I discussed its potential uses
- Research into what sensors it would need to do the different patrol modes I thought of
- I wrote part of the report about this
- I researched into different coding methods and decided to use python and opencv
- I wrote part of the report about this
- (~2 hours)

25-27/06/20

- I used fusion 360 to model the motor mounts to attach the motor to the frame
- I printed these, found they didn't work, made some adjustments and repeated this process until they worked perfectly
- I printed 8 of these (2 per motor)
- (~3 hours)

2/07/20

- I used a jigsaw to cut the chassis out of wood
- I then used sandpaper and an angle grinder to make it look neater
- (~1 hour)

3-7/07/20

- I designed the majority of the components for the spinners (Spinners, motor mounts, ramp)
- I printed these off
- (~6 hours)

9/07/20

- I assembled all the printed parts
- I attached the motors to the chassis
- I started testing the spinner contraption and adjusting it to make it start working
- (~2 hours)

Project Diary

14-21/07/20

- I spent a week adjusting the spinners, designing new 3D models, printing them and trying different methods to make a ball transit the shoot
- (~5 hours)

22/07/20

- I found a raspberry pi camera mount on "thingiverse"
- I then adapted this on fusion 360 so it would fit to the robot
- I printed this
- (~1 hour)

24/07/20

- I stiffened the frame using a beam of wood at the front
- After adjustments tennnis ball now able to reliably go up the shoot
- (~4 hours)

25/07/20

- I painted the chassis
- I used an angle grinder to slant the tray mounts so the balls in the tray would go to the back
- More adjusting the spinners mount
- (~4 hours)

26/07/20

- I put all the electronics together and tested the robot driving ability for the first time
- (~2 hours)

28/07/20

- The power distribution board was not working so I soldered a new one on
- I added a switch
- I mounted this to the robot
- (~2 hours)

10/07/20

- To replace the 2 motors at the back, I added the caster wheel
- (~1 hour)

27/07/20

- I used "InDesign" to start all the documentation to go in the report
- (~5 hours)

4/09/20

- I made 2 guides and their attachments to the robot
- I added the ultrasonic distance sensors and spent ages wiring these neatly
- I tested these but couldn't get them to work
- (~6 hours)

Project Diary

4/09/20

- I purchased new wheels/adapters. Existing axle/wheel interface is failing
- I attached these
- I elevated the motors using wooden shims to get the right floor clearance
- I attached the battery properly
- I got the distance sensors to work(The issue was that the wiring was wrong)
- I got the basic opencv code to detect balls
- I presented the current robot to neighbors and got some feedback
- (~8 hours)

5/09/20

- I tried to make my own IR sensors
- I reviewed and improved the code
- I did some more work on the report
- More testing
- I attached a new bigger caster wheel – thr original was too small to respond to static turning commands
- (~6 hours)

6/09/20

- I resoldered the wires for the battery as I had to move the switch to improve the wiring layout
- I made a platform for the switch
- I made all the wiring around the back a lot neater
- I soldered all the wires for the IR sensors
- (~6 hours)

7/09/20

- I attached the IR sensors
- I did a lot of work on the project report
- (~4 hours)

3/09/20

- I spent 3 hours coding
- (~3 hours)

16/09/20

- I took the robot to a tennis court and adjusted the code
- (~3 hours)

25/09/20

- More adjusting of the code
- Lots of work on the report
- (~5 hours)

Total hours = 82

Conclusion

I was really pleased with this result. The robot was an actual help to tennis players which was what I aimed to do. On the other hand, there is definitely more things I could have done if I had, had more time/resources. E.g. To make it more user-friendly, you would be able to control it through an app or possibly the micro-bit and it would park itself in the middle by a net post when it has enough tennis balls. Also, it obviously needs more optimization.

Wider implications

This robot would allow disabled people to access tennis. I think it is currently successful in doing that however it would be better if, when it got enough balls, it somehow sensed the location of the disabled person and brought it to them.

Currently it would be quite difficult to manufacture the robot industrially so to make this more accessible I would need to design a design that would be easier to manufacture. I could plasma cut the chassis out of metal, design a custom control board instead of using a raspberry pi and mold the plastic parts instead of printing them.

What I have learnt

I have learnt a lot during the project. I greatly improved my fusion 360 rules, woodworking skills, soldering skills and computer vision skills. I also learnt about project management and the importance of keeping a careful record of development.