UNIVERSITY OF CAMBRIDGE

Department of Engineering

TAIPEI101 (M101)

First report

Name	College	Lab group
Laura S. Clapham	Emmanuel	43EM
Ben L.C. Silva		
Caleb C.C. Chan		44EM
Tom Hill		
Derek J. Dong		45EM
James C-C. Lecomte		

I. Approach for solving the problem

Before specific engineering solutions were proposed, the whole project was divided into several core functions. Various solutions for each function were then presented, discussed and evaluated.

The most challenging problem is searching for the blocks. We devised two solutions, but both encountered problems. Though we have fitted the sensor curve with MATLAB [**appendix A1**], the error between calculated distance and real distance is still large. The method of using two ultrasonic sensors [**appendix A2**], requires a long distance between the sensors to get the best results, however, the distance between two sensors results in a triangular area in front of the robot that cannot be searched. All existing options have obvious drawbacks, but this problem is considered not insurmountable but resources consuming. More detailed discussion can be found in electronics part.

The table below summarises pros and cons for each solution. The red solutions are selected for further development (i.e. our approaches for solving the problem), and the bold pros and cons are important reasons for our choices.

Core functions	Solutions	Pros	Cons
Moving	3D printed Mecanum wheel	- Flexible, can move in 360 degrees and rotate in place	- Hard to build and test - **Only 2*40RPM + 2*18RPM motors are provided, while Mecanum wheels prefers 4 same motors**
	2 big active wheels + 1 steel ball	- **Simple structure** - Save 2 motors	- More cumbersome than Mecanum wheel
Blocks Searching	2 ultrasonic sensors scan together. Distance and direction of the block can be derived from 2 distances data	- Can be an **alternative solution**	- Complicated algorithm to get direction - A triangular area cannot be searched
	1 rotating IR distance sensor. Scan the whole square and find blocks	- **Easy to implement in algorithm**	- Hard to convert voltage measured from sensor into distance
Metal detecting	Coil metal detector	- Can be combined with arm	- Difficult to calibrate to our situation - Complicated physics
	2 coil system	- Very simple circuit design and physics	- **Wasn't effective when testing, so only worked in theory**
Navigating	Line-follow	- **Easy in algorithm**	- Unknown global position, hard to get back once lost
	Camera & CV	- Most powerful	- Very complex algorithm - **Out of control, no teammate has relevant experience**

II. Overall System Demonstration

See System Level Block Diagram at [**Appendix O1**].

The Arduino will constantly read data from the line sensors and distance sensor which it will feed to the navigation and object detection sub routines. When there is no object detected, the robot will proceed to the block area, when it reaches the box, it will store its last know location, so it can navigate back to the line from within the box. **The robot will rotate on the spot until it detects a block.** When an object is detected, the robot will navigate to the block and then start the grabbing cycle. The raising motor will lower the arms and metal detector, and then close the arms. The metal detection algorithm will wait for an increase in inductance to be observed from the search coil, if no increase is registered the navigation subroutine will be told to go to the disposal zone for non-metal blocks. If an increase is registered, it will proceed to the metal block zone. Currently there is no plan to use the network interface for computer vision etc.

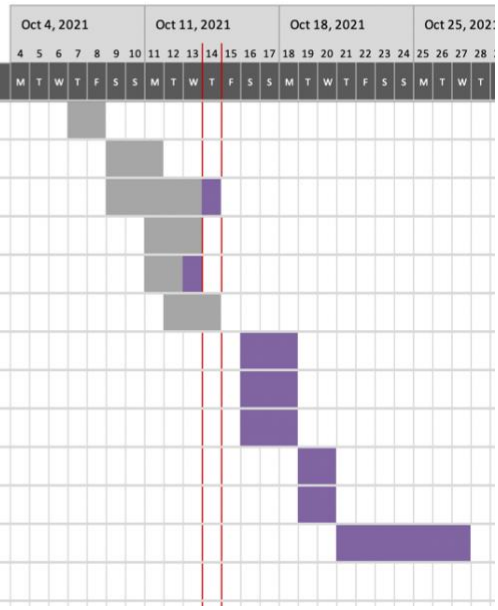
III. Gantt Chart

TAIPEI101

Project Start: Thu, 7 Oct

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Define goals	ALL	100%	Thu, 7 Oct	Fri, 8 Oct
Schematic Diagram	ALL	100%	Sat, 9 Oct	Mon, 11 Oct
Test all components, provide drive code	Derek	90%	Sat, 9 Oct	Thu, 14 Oct
Draw line-follower circuit	Derek	100%	Mon, 11 Oct	Wed, 13 Oct
Draw metal-detector circuit	Laura	80%	Mon, 11 Oct	Wed, 13 Oct
Build line-follow robot	ALL	100%	Tue, 12 Oct	Thu, 14 Oct
Build metal detector	EE	0%	Sat, 16 Oct	Mon, 18 Oct
Block searching mechanism	EE & Software	0%	Sat, 16 Oct	Mon, 18 Oct
Build grab system	ME	0%	Sat, 16 Oct	Mon, 18 Oct
Assemble all parts (consider your own task to assemble robots and add below this task)	ALL	0%	Tue, 19 Oct	Wed, 20 Oct
Draw overall circuit diagram and build circuit on Proto Board	EE	0%	Tue, 19 Oct	Wed, 20 Oct
Start test	ALL		Thu, 21 Oct	Wed, 27 Oct



IV. Technical details

a. Mechanical

Design of Grabbing Section

After deciding to use a metal search coil to detect the metal content of the block, we thought it would be possible to lower this coil over the block and use it to drag the block around. We tested this idea but found dragging the block up and down the ramp was not very reliable. Therefore, we decided to construct a grabbing arm that could lift the block to provide clearance on the ramp. To achieve this, we will have a front section connected to the main chassis by a rotating rod that can be raised and lowered by a string connected to a servo motor [**appendix M1**]. Mounted on the front section will be a small motor and 2 arms. If the small motor directly drove the arms, it would need to be standing up vertical, which would be very unstable and difficult to construct, so to drive the motors we decided to place it in the same plane as the gears and connect it via a worm gear [**appendix M2**]. After constructing a cardboard prototype, we decided this design was unsuitable, as to turn one notch on the arm gears would require one full worm revolution, and as the small gear has only 18rpm, to turn approximately 6 notches to close the arms would take 20 sec. As we only have 5 minutes, we decided 40 seconds to go through the opening and closing cycle was too slow. To replace the worm gear, we will use a helical gear mounted onto the arm gear [**appendix M4**]. This will reduce the time to around 1 second.

The front section will be made from laser cut 6mm MDF 200mm x 80mm. The brackets connecting the motor and gears will be handmade steel brackets, and the front section will be connected to the main chassis by a steel plate spot-welded to the rotating rod. The arms will be prototyped in bent steel, and then 3d printed if necessary [**appendix M3**]. The tips will be made coarse/be applied with heat shrink to increase friction.

Main Chassis design

With the grabbing mechanism decided, our focus shifted to the main chassis which the mechanism will be attached to. As the arms will continuously grab the block and be lifted until the robot reaches the collection area, there is no need for a place on the chassis to store the block. Therefore, in order to keep manufacturing process as simple as possible and high accessibility to components, we went with a flat platform design. The chassis is made of 6mm MDF sheet laser cut to 250mm x 200mm. When deciding on what wheels to use, we discovered it would be difficult to steer the robot if we used 4 wheels. Therefore, we decided to go with a tri wheel system, which uses two motorised wheels and a ball caster as the third support. This allows us to steer the robot by just giving the motors different angular velocity as the ball caster is free to swing left and right. The 4" wheels are chosen for two reasons: a) to maximise speed (40rpm x 4" x π = 12.8 m/min, with the distance between start area and collection area being 2.25m, the total ideal travel time of the robot will be 2.1 minutes for 6 round trips, which is well under 5 minutes) and b) to keep the chassis high off the ground enough so it doesn't get stuck on the edge of the ramp and the bridge [**appendix M5**]. The ball caster will be mounted onto the chassis through a long arm in order to keep the chassis levelled.

In order to raise the front section, a servo motor is mounted on the chassis with the rotor being on the same distance above the chassis as the length of the mechanism board. It will pull a string attached to the front section, which will pivot it against the front of the main chassis. The max torque required is calculated in [**appendix M6**], when the board is lowered slightly below horizontal. Shelves are placed at the rear of the chassis to place the Arduino boards and circuit boards. In order to keep robot steady on its supports, the battery is mounted towards the rear of the chassis offset the moment created by the front section. The mounts will be fabricated by bending aluminium sheets into the right shapes, except for the motor mounts, which will be 3D printed for extra precision in order to grip onto the motor properly. A CAD model of the main chassis is shown in [**appendix M7**].

A prototype chassis is made to let the electrical and software teams to test their designs on. It is made from cardboard cut to the same size as the original design, while all the mounts are made with aluminium sheets [**appendix M8, M9**]

b. Electronics

Line following

OPB704 line sensors will be located underneath the chassis of the robot. The LED will emit IR radiation and if the robot is over the white line, the radiation will be reflected back to the phototransistor. The turns on the transistor and provides a path straight to ground. If the OPB704 is over the white line the phototransistor will input a low value to the inverter which will input a high value to the Arduino and if it is over the black part of the floor the opposite will happen.

There are 2 resistors connected in the line following circuit **[appendix E1]** R1 and R2. Some simple calculation shows that 62.5 Ohm is the most suitable value, and we choose a 68 Ohm resistor here. For R2, an experiment was done to determine the best values: the voltage at the output from the phototransistor was measured when sensor was directed towards a white line. Since the output should be LOW (0 voltage ideally), the lower the voltage the better. The results are tabulated in the appendix

[appendix E2] and it was found that higher values for R2 work better and thus we choose a 1M ohm resistor.

We chose to use an electronic method (i.e. an inverter chip) to convert analog signal from the line sensor OPB704 into digital signal, rather than setting thresholds in algorithm or even directly read analog signal *by digitalRead()*. The explanation for this is that the application of a inverter makes the values read by the Arduino more stable.

Detection of blocks

Originally the HC-SR504 ultrasonic sensor was going to be used to detect the blocks. It sends out sound waves and detects the time it takes for them to return. This is sent to the Arduino and in the code, this can be converted into a distance. If the distance detected by the HC-SR504 at a specific angle is different from the expected distance (as measured before hand), a block has been detected and the robot can stop rotating. A problem was recently discovered with this approach as the ultrasonic sensor has a large response angle meaning it will not be able to detect the direction of block accurately **[appendix E3]**.

The GP2Y0A21YK0F infrared distance measuring sensor was trialled to see if it could replace the HC-SR504 but it was found that its output was very noisy and it gave different results for different objects (although this shouldn't be too much of an issue as both the objects we want to detect are covered in the same material).

Another solution could be to use two ultrasonic sensors at different sides of the robot and trigonometry to find the direction to the block. Detailed formula to calculate direction can be found in **[Appendix A2]**.

Metal detection

This seems to be one of the most challenging sections of the project as many designs for metal detectors require an in depth understanding of inductors and capacitors and have many different components that may be not working as hoped. Our first ideas when thinking about how the metal and non-metal blocks could be detected included measuring differences in weight, using a magnet to attract the metal block (movement detected by a distance sensor) or to use the properties of magnetic fields. The former two haven't been tested at all due to assumptions that the differences in weight are very small so wouldn't be able to be detected and that the magnet wouldn't be strong enough to attract the block a detectable distance. A design involving two coils was tested and was found not to work. This design (and all subsequent designs) relies on the fact that when metal comes close to a coil, its acts like an iron core and increases the inductance of the coil. The theory was that when the coil was placed over a block the robot would be able to tell if it contained steel or not because if it did contain steel, the inductance of the coil would increase, causing a change in the magnetic field, inducing a voltage across the second coil due to Faraday's law. This voltage could be amplified and input into the Arduino. When a prototype **[appendix E4]** was built, no potential difference could be measured between the two ends of the outer coil.

Other unsuccessful prototypes using the change in inductance principle include the use of oscillators. If the inductance of the coil increases, the resonant frequency of the circuit including the inductor and capacitor changes.

The present design for the metal detector places the coil as part a high pass filter as shown in the circuit diagram **[appendix E5]**. A block wave is fed into the circuit. When the current rises, the inductor creates a magnetic field that causes an opposition to this change in current, so more current flows to the capacitor (as less current flows down through the coil), so more charge is stored by the

capacitor and the voltage over the capacitor will be higher. The diode ensures that the capacitance cannot discharge. Over a number of pulses the voltage over the capacitor can be measured. If there is metal close to the coil, the inductance of the coil increases therefore so does the voltage measured over the capacitor. The Arduino will pick up this increase in voltage and will change the output connected to an LED to HIGH, illuminating the LED.

It is easier to use software and digital signals to illuminate the LED rather than the changes in voltage across the capacitor as the difference in voltage in the electrical side may not large enough and may vary from one test to the next making the system unreliable.

So far, the Arduino can detect changes in the voltage over the capacitor but the outputs to the LEDs are not reliable. The next tasks will be to make changes to values in the code and to perhaps change values of the components, such as the capacitors, or the dimensions of the coil. Moreover, another LED needs to be illuminated when the plastic block is found.

A block diagram is provided in the appendix to show how the metal detector and line sensors work **[appendix E6].** Our overall circuit diagram **[appendix E7]** is also shown in the appendix.

c. Software

Main program structure
The program consists of subroutines that enable navigation with line following, object detection with the ultrasonic sensor, metal detection, object grabbing and lifting, placing objects into the appropriate boxes, and recovering from the loss of a line.
All these subroutines are defined before the main loop of the program.

The behaviour of the robot at any given time will be state-driven; at any arbitrary time-step, using a combination of current memory state and inputs via the Arduino pins, the robot will modify its state and actions, calling subroutines to do so.

In order to allow easier integration with the electronics and mechanical aspects of the project and be more flexible to changes in design, we will aim to abstract the purely algorithmic parts of the software. The program will communicate with the hardware through an interface of modular methods which can be swapped out and replaced to fit changes in specification, minimising the need for large changes to the underlying architecture.

Parallel processes
Since for most of the time the robot will be carrying out continuous tasks while also constantly pulling sensor data and checking conditions for state changes, it's important that these processes run in parallel. A subroutine shouldn't block the execution of the program by running for a non-trivial period, for example by using delays in a for loop for a continuous motion, as this essentially blinds the robot during the execution. For critical continuous processes we will instead separate out the step of the process that is repeated into its own procedure and call that in the main loop of the program, allowing parallel processes.

Since main-loop processes may require some form of state in the global scope (i.e. previous motor speed for an acceleration method), we will aim to only extract methods in this way where necessary. (See **[appendix S2]** for a method which currently blocks processing for >=130ms, and may be extracted into the main loop if testing shows this to be a liability)

Line Following
The two IR line sensors are positioned either side of the line so when neither sensor detects the line the robot will progress forward. If the left sensor sees the line, then the robot will make an adjustment with a left turn until neither sensor detects the line and vice versa for the right sensor. A left turn is made by driving the right motor forward and driving the left motor backward **[appendix S1]**. There has been a discussion about whether two sensors are sufficient for reliable line following as the inputs are the same if the robot is on the line or it has lost the line completely. However, with this algorithm the robot will be able to recall its recent movements and correlate these with the inputs from the sensors. For example, suppose the robot starts on the line and moves forward. If the left sensor detects the line, it can be inferred that the robot is straying to the right. The robot should then turn slowly to the left until the left sensor no longer detects the line and thus the line must be in between the two sensors. On the other hand, if the robot were to go straight forward instead and the left sensor again loses the line, it is inferred that the robot has gone off the line.

Object detection
The team intends to use an ultrasonic sensor to monitor if there are any objects within the robot's line of sight. Once the robot reaches the collection box (detected by the line sensors both being activated) the program will enter an exploration subroutine. This subroutine will pivot about the box entrance and wait for a distance reading within the range of the box to appear on the ultrasonic sensor at which point the robot will stop and move forward by the appropriate distance until a box is reached. If the robot fails to detect a box on the first sweep, it is told to advance further into the box and repeat the sweep. The first box has a predefined position at the box entrance so the first iteration of the object of the object detection system does not require a sweep sequence and can simply enter the grabbing subroutine.

Grabbing arm
The servo motor for the lifting and lowering of the grabbing arm will be attached to pin 10 on the Arduino and interfaced with through Servo object methods. Subroutines will be called when lifting and lowering the arm; the Servo object will be written to iteratively in a for loop, with values corresponding to the angles to turn through as the servo rotates. Since the robot will not be moving or responding to inputs during this period, it is acceptable for the procedures to block execution of the program while running. Therefore, the entire movement can be done in one pass of the subroutine using delays for continuous movement, as opposed to having to iterate in parallel to other main loop behaviours over the course of many program loops.

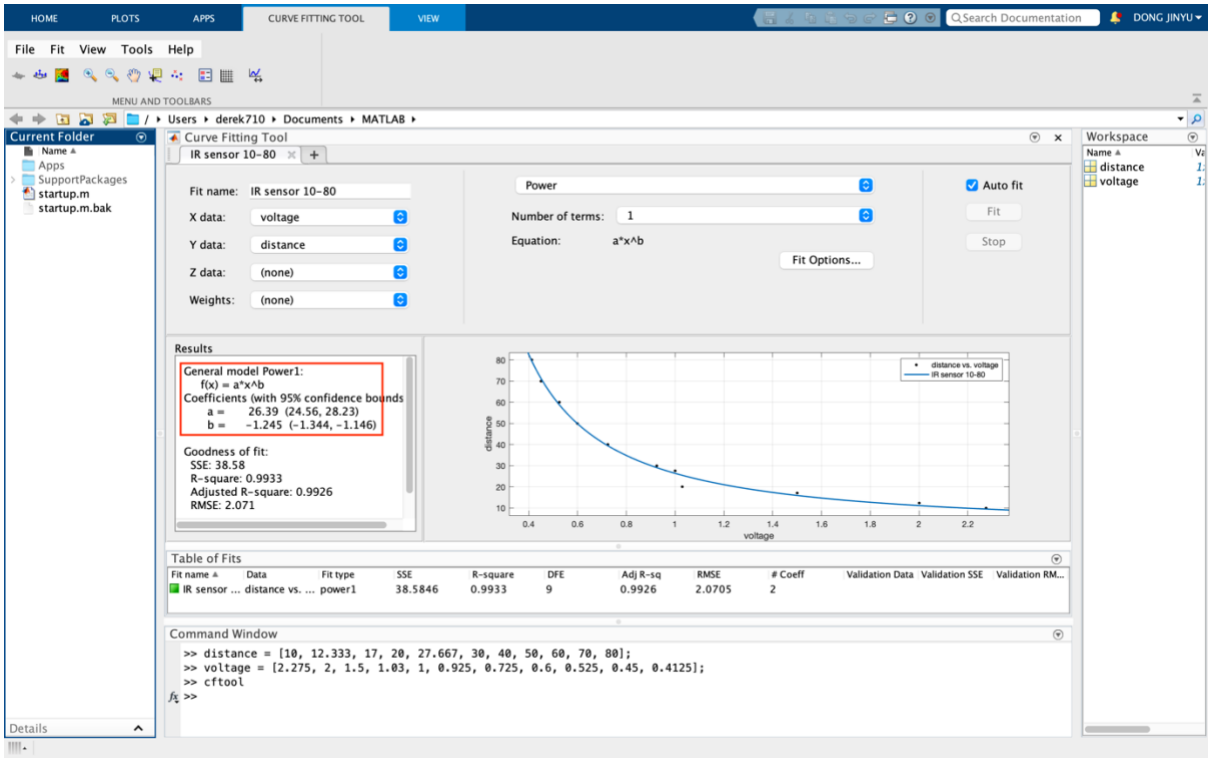
Reading sensor values
When reading sensor values like the line sensors under the robot, the readings can quickly fluctuate and in the noise supply incorrect values. To get a usable reading from sensors, we will add a layer of abstraction using an interface to access sensor values with a rolling average.

During each main loop of the program, for each sensor we will read the value of the respective input pin. We will push this value to a variable of an abstract data type which acts as a constant-length FIFO queue array, with length of ~10 values. Ideally, we want to have a range of sensor readings over a time-period of 10-200ms, since too long a sample period makes the robot less responsive to discontinuous changes in sensor readings, and too low a sample time can allow irregular readings to give false positives.

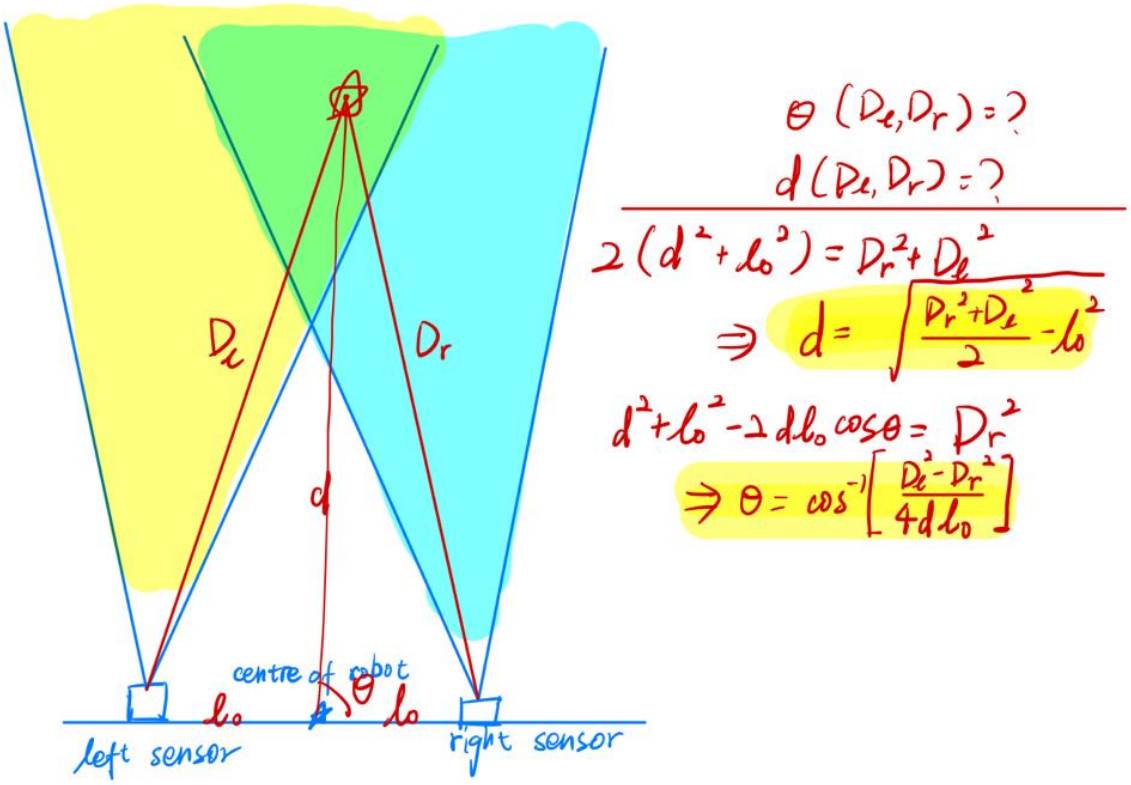
The interface for parsing and reading the sensor's value will be provided as a function that will be called in the program when a read is needed. The function will return a rolling average of the sensor's value by returning the mean of the sensor's array values. For digital sensor inputs, their interface function will return True or False depending on whether the rolling mean is majority high or low.

Appendix

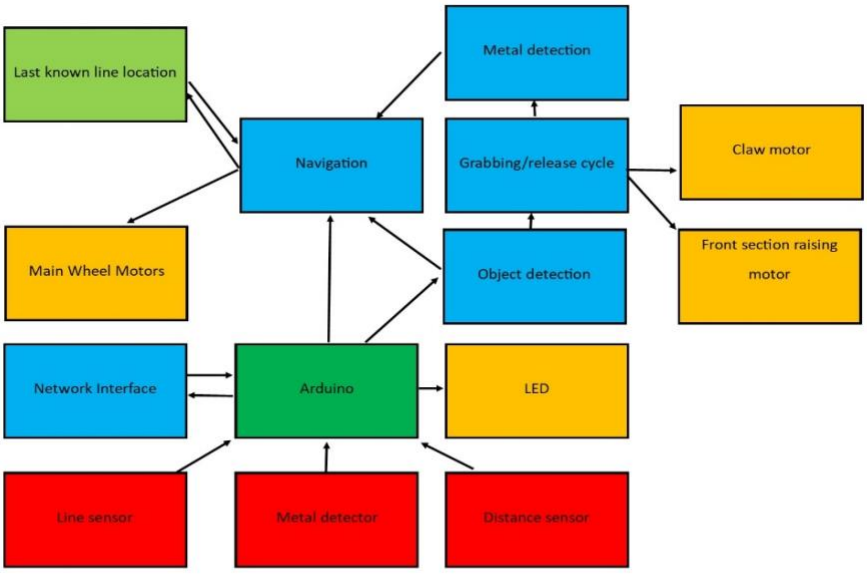
Appendix A1: Curve fitting in MATLAB



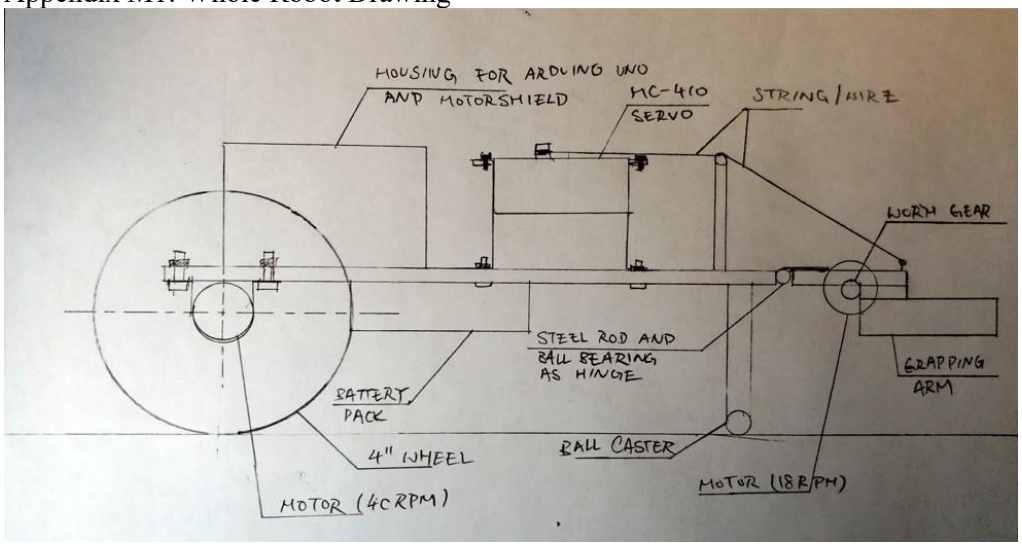
Appendix A2: Searching blocks with 2 ultrasonic sensors



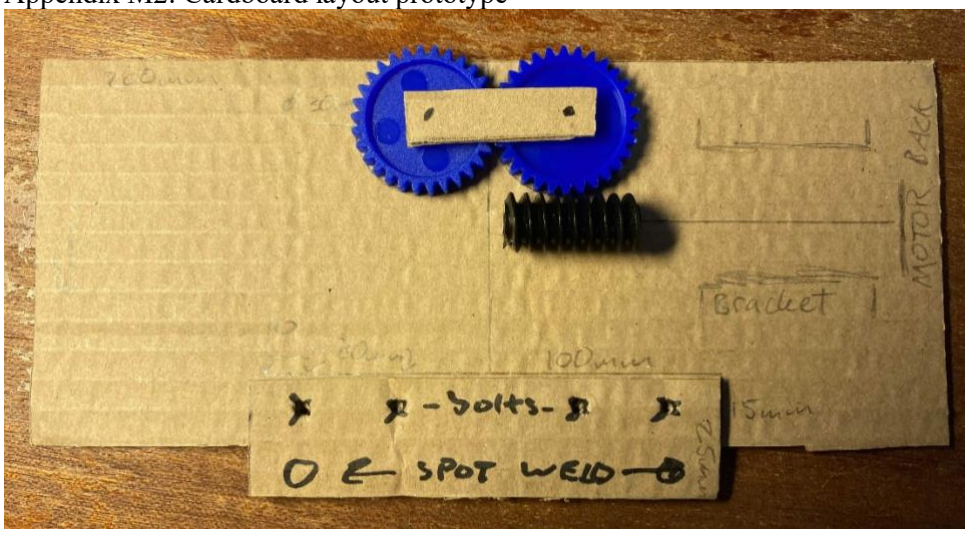
Appendix O1: System Level Block Diagram



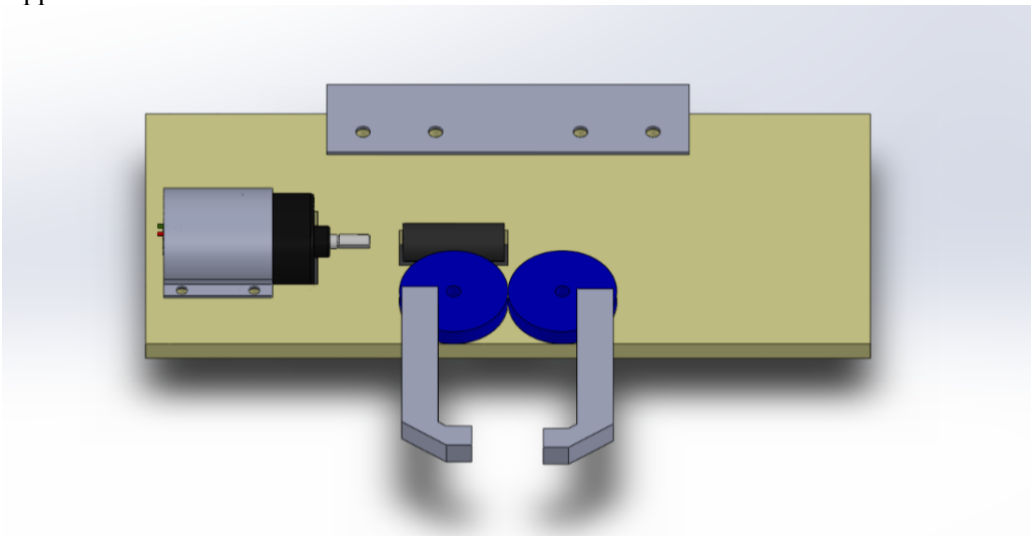
Appendix M1: Whole Robot Drawing



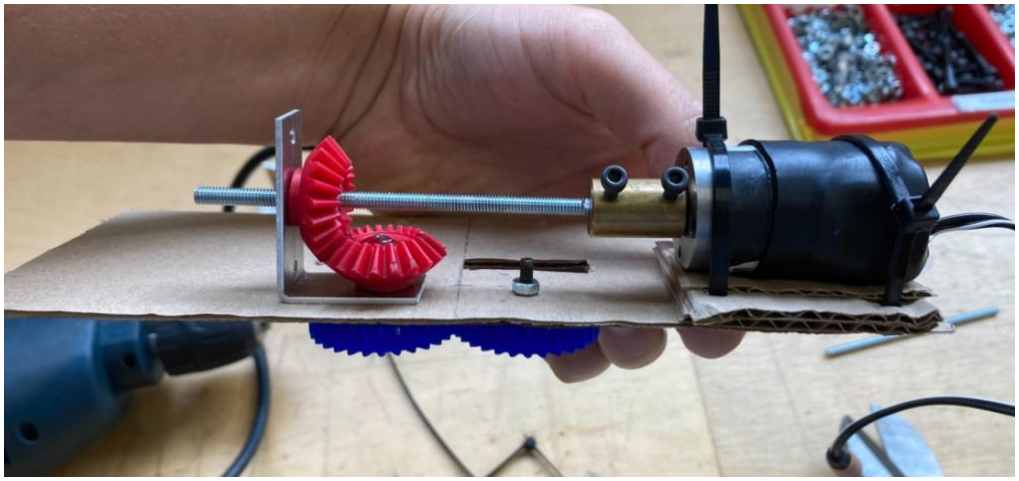
Appendix M2: Cardboard layout prototype



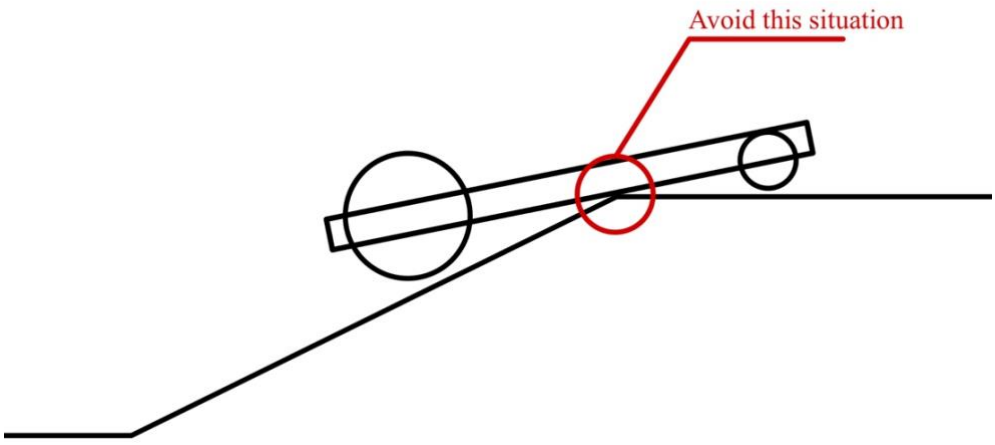
Appendix M3: Solidworks CAD render



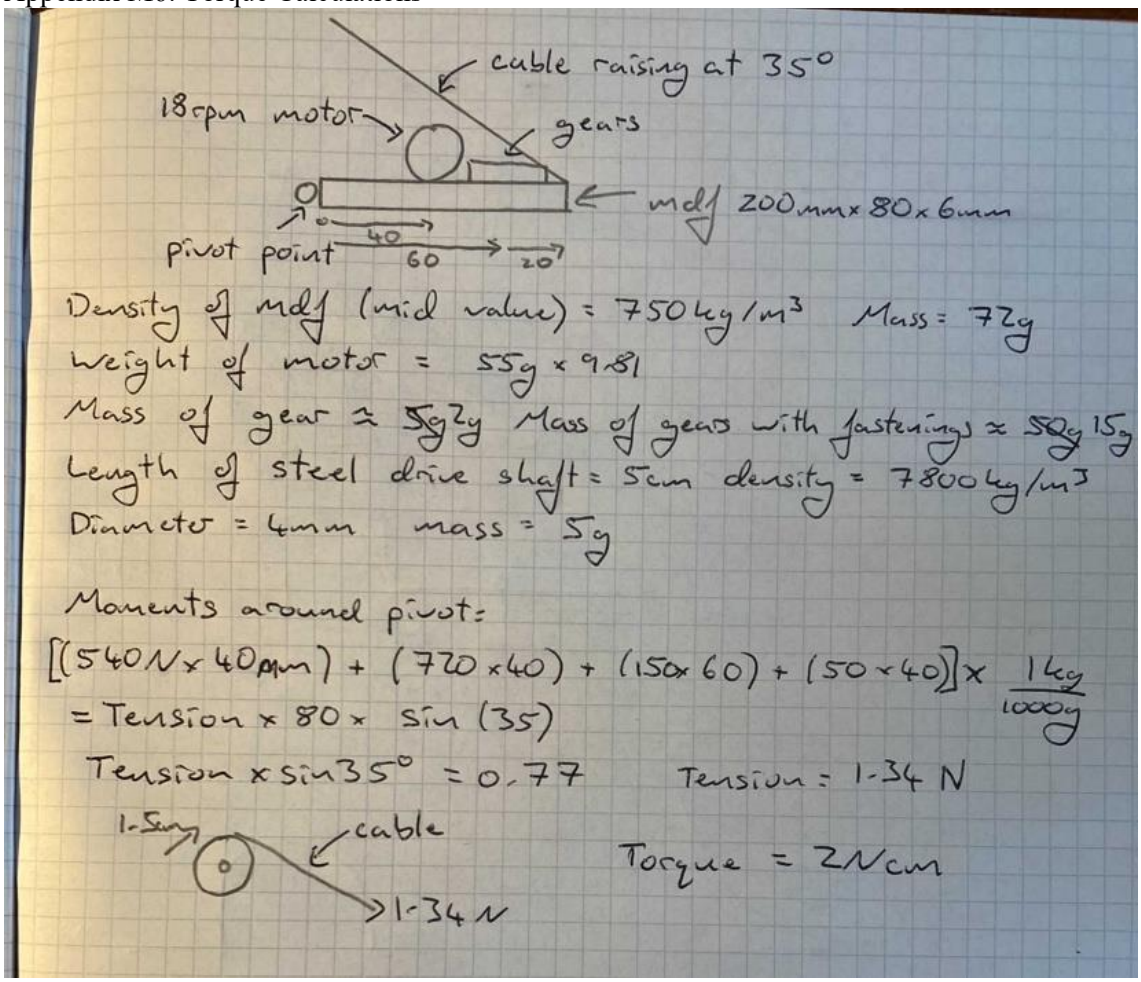
Appendix M4: Helical gears connected to motor



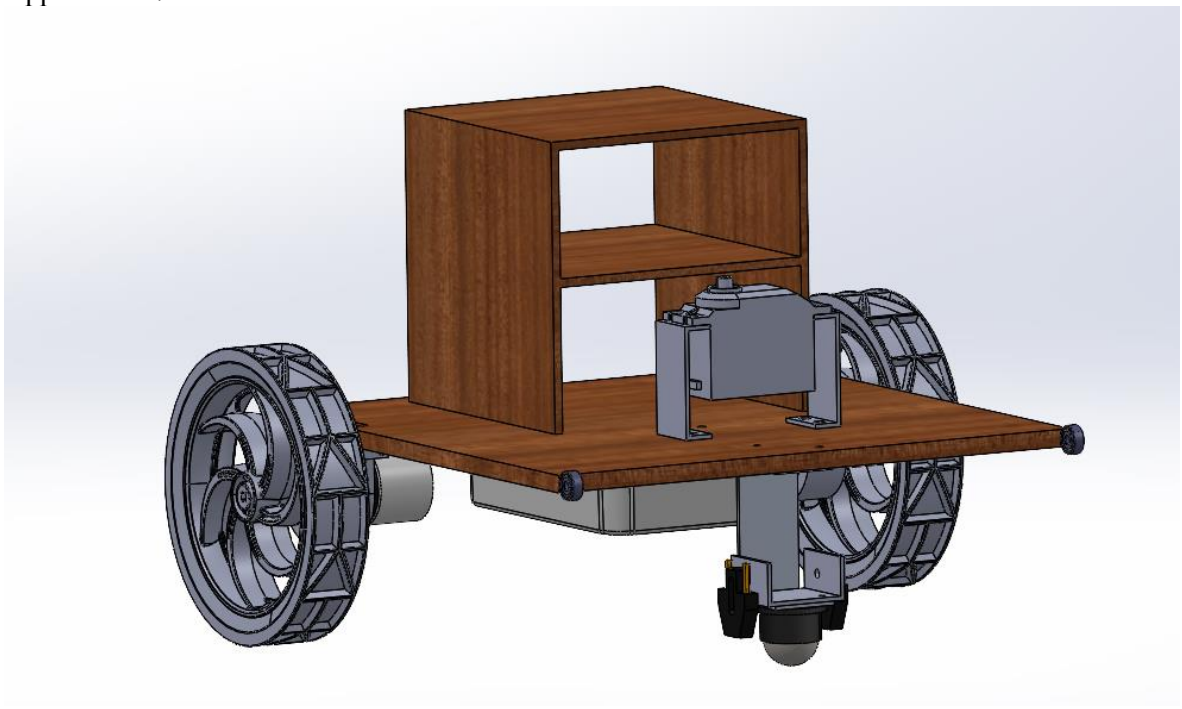
Appendix M5: Reason (b) for using 4’ wheels



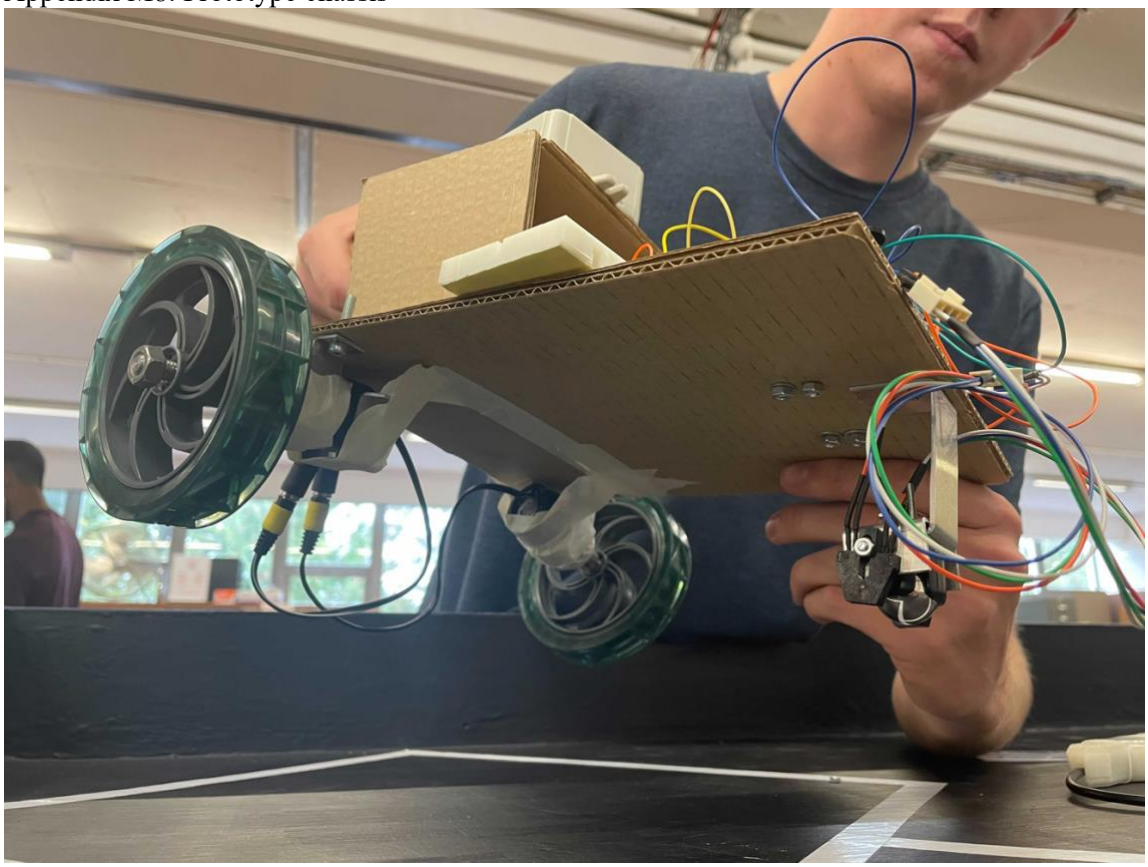
Appendix M6: Torque Calculations



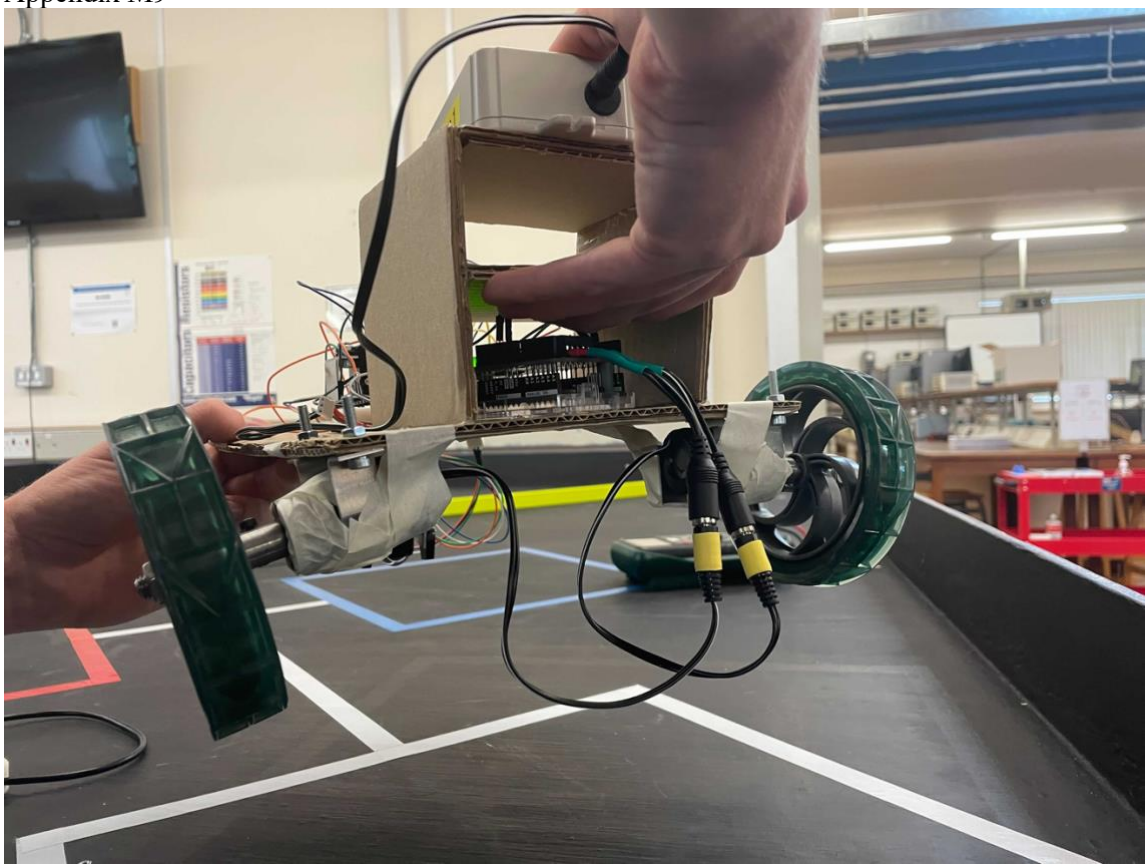
Appendix M7: Solidworks CAD render of robot



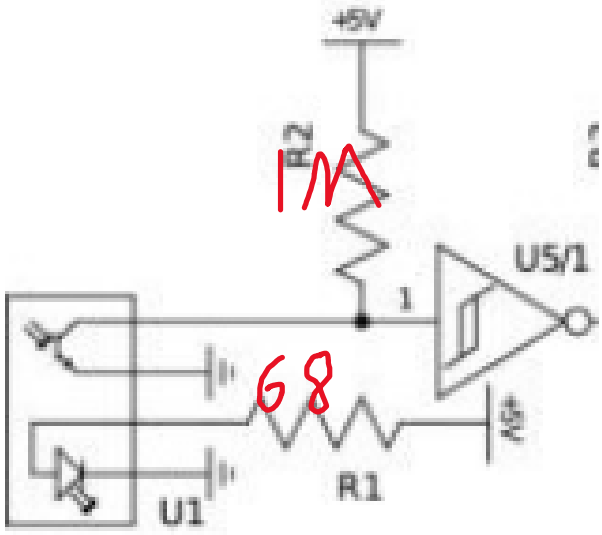
Appendix M8: Prototype chassis



Appendix M9



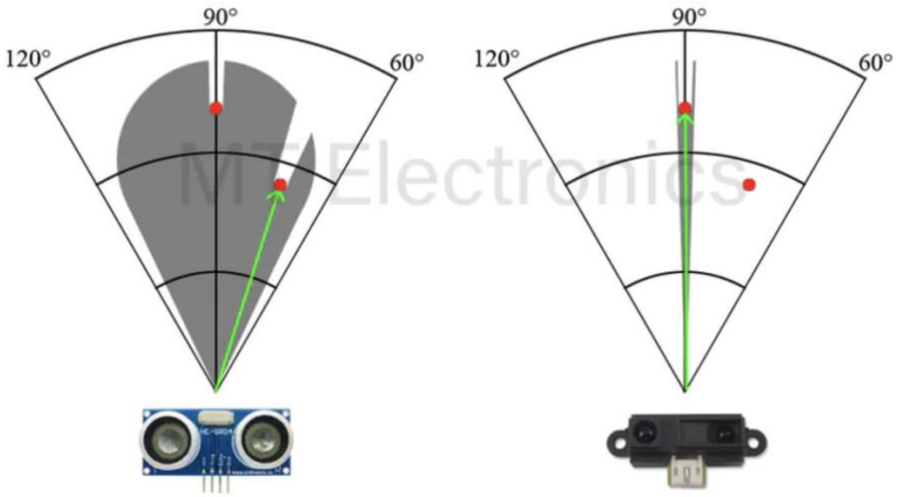
Appendix E1: Line follower circuit.



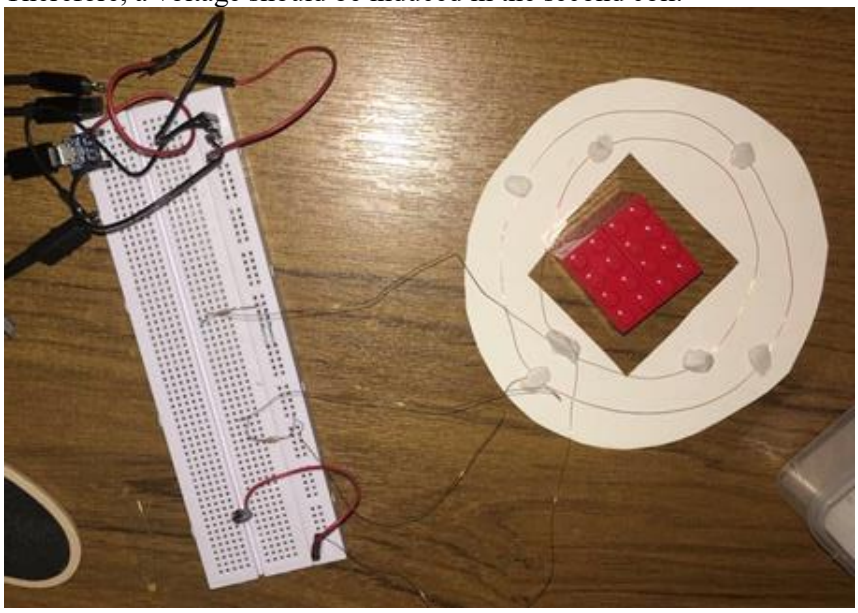
Appendix E2: Table showing how the resistance of R1 affects the output from the phototransistor.

Resistance of R2 (kΩ)	Voltage from phototransistor on white line (V)
36	2.5
66	2.2
68	2.0
108	1.2
170	1.2
1000	0.05

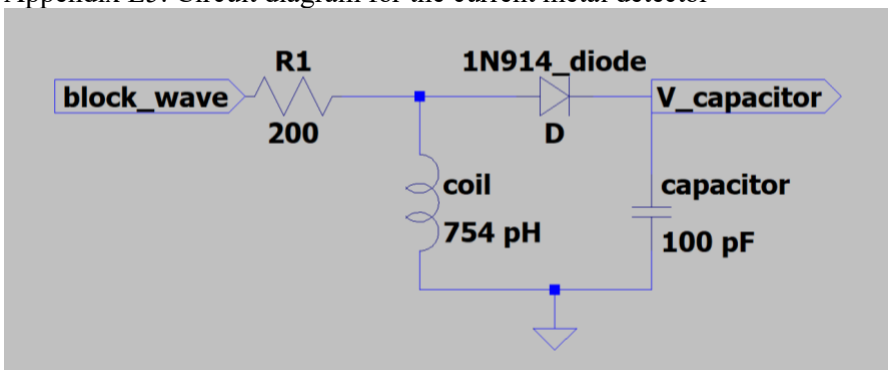
Appendix E3: Diagram showing the difference in response angle between the ultrasonic and infrared distance sensors.



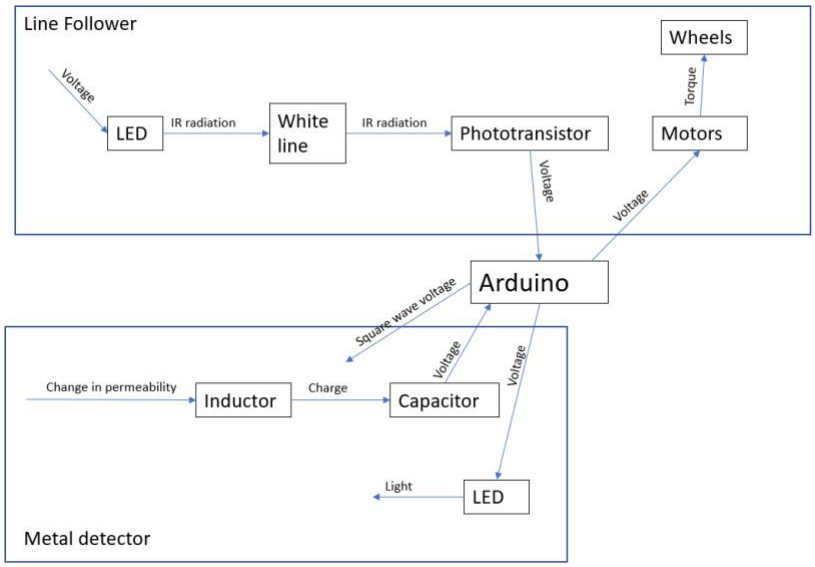
Appendix E4: Prototype that didn't work.
The inner loop is supplied with a DC current from the 5V supply from a USB. The ends of the outer coil just connected via a resistor. Theoretically when the metal block, in red, is placed inside the coil, there should be a change in the magnetic field caused by the inner coil due to its change in inductance. Therefore, a voltage should be induced in the second coil.



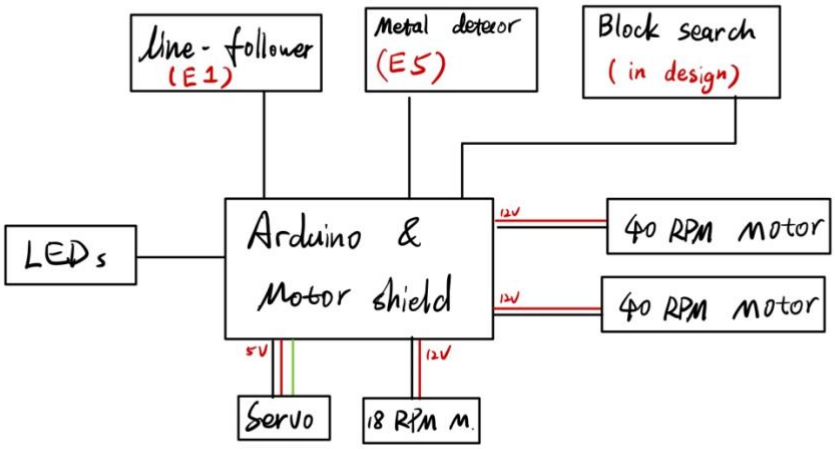
Appendix E5: Circuit diagram for the current metal detector



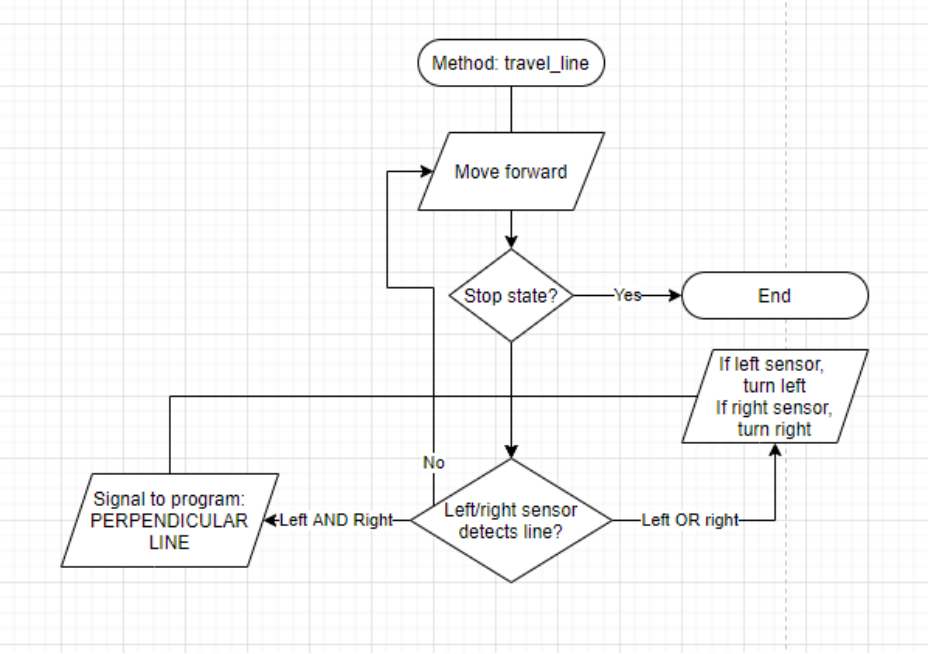
Appendix E6: Block diagram to show how the metal detector and line sensors will work



Appendix E7: Overall circuit diagram



Appendix S1: Basic flowchart for line following algorithm



Appendix S2: Procedure for acceleration from a stationary state into turning right. Since this is a fast process, the iteration has not yet been extracted into the main execution loop, however further testing will be done to assert if this is necessary.

```
void right(){
  int TargetSpeed = 255
  leftMotor->run(FORWARD);
  rightMotor->run(BACKWARD);
  for (int MotorSpeed = 0; MotorSpeed <= TargetSpeed; MotorSpeed = min(MotorSpeed + 10, TargetSpeed){
    leftMotor->setSpeed(MotorSpeed);
    rightMotor->setSpeed(MotorSpeed);
    delay(5);
  lastCall = "right";
  }
}
```