Group U Robot Project Report

ENGR204

By Alex Reynolds, Ben Draper-Turner, Duncan Lancaster, Matt Dean, Sherwin Robinson

Table of Contents

1 Introduction 2
2 Comparison to Initial Report 2
3 Coding 3
3.1 Original Plan 3
3.2 Line Sensing 3
3.3 Arm Mechanism 3
3.4 Main Structure 4
3.5 Final 4
4 Electronics 5
4.1 Line Sensor/ Wheatstone/Shroud 5
4.2 PCB/ amplifier 6
4.3 Motor Circuit 7
4.4 Arm Electronics 7
4.5 Wiring map 7
5 Mechanics 9
5.1 Arm Mechanism 9
5.2 Issues with Arm Mechanism 10
5.3 Chassis and Electronics Housing 10
5.4 Issues with Mechanical design 13
5.5 SOLIDWORKS Drawings 14
6 Testing 14
6.1 Evaluation of Mechanical Performance 14
6.2 Evaluation of Electronic Performance 14
6.3 Evaluation of Coding 14
6.4 Overall Performance 15
7 Conclusion 15
7.1 Areas for Improvement 15
7.2 Overall Conclusion 15

1 Introduction

The robot project is a second-year project where groups of students must design, build and test a robot. There is a specification the robot must be built to and a task it must complete. The initial brief describes the task;

The aim of this project is for each group to design, build and operate a line-following robot. The robot should navigate a course consisting of a line of tape on the floor (with the tightest radius 0.5m) and stop when it reaches a vertical wall of height at right angles to the line. Further, it should carry a closed C6 envelope containing one sheet of A4 paper and a mechanism designed and built to post it through a slot 0.2 m beyond the stopper wall.

At the end of the project, a report is to be written. This document is the report for group U. We have broken the robot down into three sections. Mechanical, electrical and software. This report looks at the design and building of each of these elements. We also evaluate the performance of each element and state any improvements that could be made.

2 Comparison to Initial Report

In comparison to our initial timescale, the project initially performed very well, with all tasks completed by their scheduled earliest completion date, and as such we were able to progress easily up until the point at which the chassis DXF. file was to be sent. There was a large delay in getting the chassis cut out due to a technical issue with the water jet cutter, and as a result we received our chassis two-weeks later than every other group. Fortunately, this did not affect the electronics progress meaning that the line following electronics were assembled and tested in time for the line following deadline (20/02/19), however the delay in the chassis meant that it was only received midway through the lab session in which the robot should be assembled and following the line. This was not enough time to assemble the electronics on the robot, and test that. We were able to receive a one week extension to this deadline, which meant the next week we successfully demonstrated the robot and passed the test, however the delay introduced by the chassis issues meant that we had effectively lost two weeks' worth of lab time in the mechanical lab, putting this aspect of the project behind schedule.

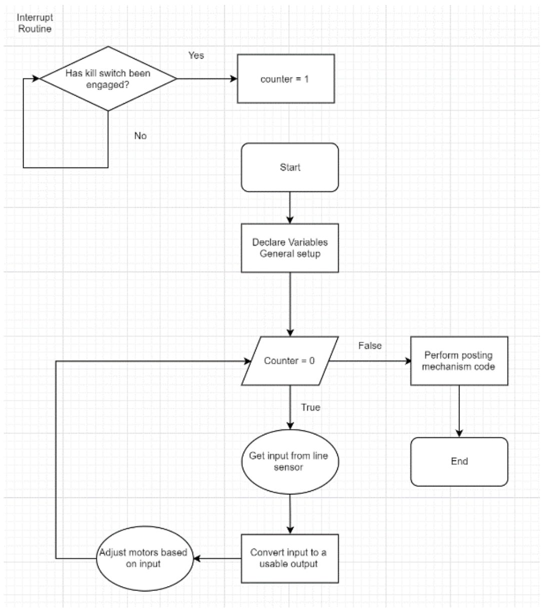
During this session whilst adjusting our project plans going forwards to account for the delays we also discovered the main issue which caused us to deviate from the initial project plan, namely that we had failed to realise that week 21 of the university term dates was in fact in the Easter term, rather than the summer term as we had thought. This brought our final deadline substantially forward, from the start of May to the 20th March, meaning that all of our project plan needed to be reassessed. Fortunately, due to the initial plan to have finished the assembly of the robot by the 30th March, we were able to make changes such that the critical path of activities was largely unaffected, however almost every task lost any float they once had. This meant that whilst we would still be able to complete the robot on time, any further complication at any stage posed a substantial risk to the final success of the project.

Our final robot was as intended in terms of the letter posting mechanism, and it is identical to the mechanism set out in the initial report. All other aspects of the initial report were followed correctly.

3 Coding

3.1 Original Plan

The robot can essentially be broken down into three parts. Following the line, stopping and posting the letter. This is how we planned to structure and work through constructing the code. The flowchart to the right shows the flow of the code.


Interrupt Routine
Has kill switch been engaged?
Yes
counter = 1
No
Start
Declare Variables General setup
Counter = 0
False
Perform posting mechanism code
True
End
Get input from line sensor
Convert input to a usable output
Adjust motors based on input

The backend of the electronics meant that the Arduino receives one signal. This signal is dependent on how symmetrical the sensor is over the line, essentially giving us a set point for when the robot is centred over the line. A number greater than the set point suggests the robot is too far to the right or the line is curving left, and a value less than the set point means the robot is too far left or the line is curving right. There are a few options open for how the code could be written. The first idea was to just directly adjust the speed of each wheel depending on the error (setpoint – input). After doing more research PID control seemed more appropriate. This method of control is very stable and efficient and seemed appropriate to use for our robot.

3.2 Line Sensing

PID stands for Proportional, Integral, Differential. It is essentially three control methods with the error from each one added together and weighted. The coefficients Kp, Ki and Kd are used to weight the errors. These values can then be tuned by experimenting with different values.

The proportional error (P) is what we originally planned to do. The error is calculated directly and in proportion to the distance away from the set point.

$$\text{error} = (\text{setpoint} - \text{actual value})$$

$$P = \text{error}$$

The Integral error (I) finds the accumulating error. This can generate a very large error very quickly.

$$I = I + \text{error}$$

The differential error (D) looks at the rate at which the robot is diverging from the line. This is calculated by looking at the difference between the current error and the previous error calculated. The greater the difference between the two errors the bigger the differential error.

$$D = \text{error} - \text{previous error}$$

Once all these errors are calculated, they are weighted and summed to give the final error.

$$\text{PID error} = Kp^*(P) + Ki^*(I) + Kd^*(D)$$

3.3 Arm Mechanism

Our posting mechanism can be split down into two parts from a coding perspective. Moving the arm into position followed by posting the letter. Our arm uses a servo motor to get the letter into position and then a motor and ribbon to post the letter. This requires the code to move the servo motor followed by running the motor for a set time. This is relatively straightforward to code as it is a series of events.

The Arduino has a library for servo motors that we used. You can find the functions and their descriptions on the Arduino website (https://www.arduino.cc/en/Reference/Servo). The basic principle behind the servo is that the Arduino sends a signal that corresponds to a specific position of the servo. The servo then moves to that position. This requires a small amount of time to complete so it is important to pause for a moment to let the motor move into position. The posting motor is then turned on for a specific time found through experiment.

3.4 Main Structure

Referring to the flowchart in 3.2 above3.1 above, the flow of the code needs to be:

Follow the Line – Stop at the wall – Move the servo motor – Run the posting motor.

Moving from line following code to posting code is achieved using a switch that is activated by touching the wall. An interrupt was set up on a pin wired to the switch. When the pin detects a logic change it switches the code. The interrupt works by incrementing a counter. The value of this counter decides which part of the code to run by using a series of if statements.

3.5 Final

```
/*
 * Lancaster University Robot Project
 * Final Code - Group U - 18/03/19
 * Author - Benedict Draper-Turner
 */

#include <Servo.h>
int count = 0;
int P = 0;
int I = 0;
int D = 0;
int Kp = 1;
int Ki = 0;
int Kd = 6;
int error;
int adjustedError;
int previousError;
int PIDval;
int senseInput;
int setPoint = 0;
int motorSpeed = 80;
int servoAngle = 90;
void calculatePID();
void calculateError();
void motorControl();
Servo theServo;

void setup() {
  EIMSK = _BV(INT0);                          //eneables interrupts for pin 2
  EICRA = _BV(~ISC01) | _BV(ISC00);           //sets iterupt to occur on any logic change on pin 2 (switch)
  DDRD = _BV(PD7) | _BV(PD6) | _BV(PD5);      //sets pin 7(arm motor) , 6 (motor) and 5 (motor) as outputs
  DDRC = _BV(~PC3);                                                //sets A3 (light sensing) to an input
  DDRB = _BV(PB3);                                                 //sets pin 11 (servo) as an output
  TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM01) | _BV(WGM00);   //sets up fast PWM mode
  TCCR0B = _BV(CS01) | _BV(CS00) | _BV(~CS02);                     //sets prescaler of 64
  OCR0A = 0;                                                       //sets motor control to 0 to start
  OCR0B = 0;

  theServo.attach(11);     //attatches pin 11 to the servo
  theServo.write(0);       //sets the servo to start position
  setPoint = analogRead(3); //sets the setPoint when the robot is switched on
  Serial.begin(9600);
}
void loop() {
  if(count == 0){
     calculatePID();
     calculateError();
     motorControl();
      Serial.print(setPoint);
      Serial.print("        ");
      Serial.print(error);
      Serial.print("        ");
      Serial.print(OCR0A);
      Serial.print("        ");
      Serial.print(OCR0B);
      Serial.println();
    }
```

continues on next page

```
      if(count == 1){
       OCR0A = 0;
       OCR0B = 0;
       motorSpeed = 0;
      theServo.write(servoAngle);          //moves arm into position
      delay(1000);
      PORTD = _BV(PORTD7);      //posts letter
      delay(720);
      PORTD = _BV(~PORTD7);
      count++;
         Serial.print("3");
         Serial.println();
        }
        if(count >= 2){
          Serial.print("4");
          Serial.println();
          }
  }
  void calculatePID(){                //calculates the PID error value
    P = error;
    I = I + error;
    D = error - previousError;
    PIDval = (Kp*P) + (Ki*I) + (Kd*D);
    previousError = error;
  }
  void calculateError(){              //calculates the current error from the centre of the line
                                      //and converts it into a value the PID can use
   senseInput = analogRead(3);
   error = setPoint - senseInput;
   constrain(PIDval, -motorSpeed, motorSpeed);
   error = map(error, -setPoint, setPoint, -motorSpeed, motorSpeed);
    }
  void motorControl(){            //adjusts motor speed based on error
    OCR0A = motorSpeed + PIDval;
    OCR0B = motorSpeed - PIDval;
    }
  ISR(INT0_vect){     //Interrupt service routine for when robot hits wall
    count++;          //Increases count by 1
    cli();
  }
```

4 Electronics

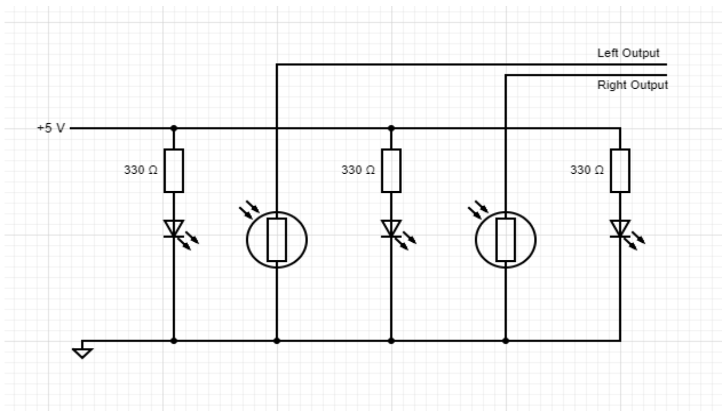
4.1 Line Sensor/ Wheatstone/Shroud

In order to design the line sensor module, we initially came up with many differing ideas for how each aspect of the design should be undertaken. The first factor was the placement of the sensor(s) however we quickly decided as a group that there should be as large a distance as possible between the drive motors and the sensor head(s) since this would allow the most accurate line sensing to occur. This is due to the fact the robot will pivot around one of the driven wheels whilst turning and so by maximising the sensors radius from this point during a turn we will allow the sensor head the maximum range of 'vision' during a curve. This location will also mean that the robot will require less of a turn to move the sensor to the desired central position on the line, meaning the magnitude of the turns required to keep the robot on a straight course will be as low as possible, leading to a smoother motion.

The design which we chose to use for the sensor head itself is that of a Wheatstone bridge, whereby two of the resistors are light dependent, which are then balanced within the bridge such that when the two light dependant resistors (LDRs) are both above the line, their resistances will match, causing the bridge to balance. if one of the LDRs is not fully above the line, or completely away from it, the bridge will be unbalanced meaning that the two outputs of the sensor will show differing voltages. This then allows us to have a sensor head with two outputs, one from each LDR, which can then be input into the amplifier circuit to give us a voltage indicating the location of the sensor head in relation to the line.

In order to give adequate light within the sensor head, 3 LEDs were added to the sensor head, powered by the Arduino. These were green, as using the plot within the electronics lecture we found that the LDRs we were using were most sensitive to this wavelength of light. This allows our sensor to be slightly more accurate than if we had used the provided red LEDs, however this difference is likely to be very small if not negligible. During our brief testing of this we were unable to find any functional difference between the Led colours

The final circuit schematic for the head is shown to the left (balancing resistors for the Wheatstone bridge are part of the amplifier circuit)



Whilst initially built and tested on a breadboard, the sensor head was transferred to a perfboard to become more permanent, and to allow easier mounting, however this first head caused us issues as the traces tended to lift away from the board, breaking the connections. Due to this a second identical head was built to replace the first, which did not share the issue leading us to believe the fault was due to a manufacturing defect in the board rather than the sensor head design.

Up until a few weeks after the line sensing deadline, we only intended to use one sensor head. However, during the building phase of the project we did consider adding another sensor head due to having all the other electronics completed ahead of schedule in order to allow a further level of accuracy. Shortly after this was the realisation that the time frames would have to be adjusted due to the week 21 oversight discussed earlier and so the revised timescale did not give us adequate time to implement the additional sensor.

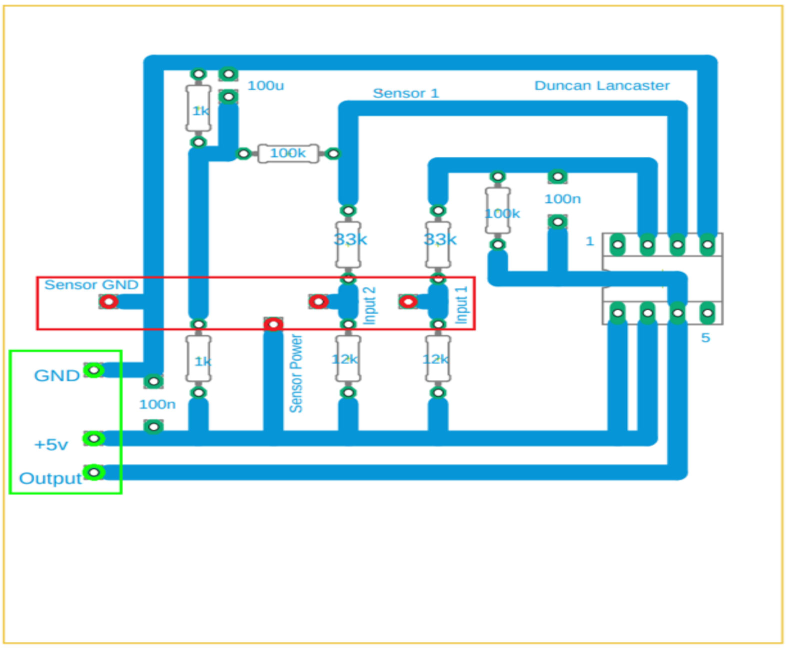
The shroud we used for the sensor head was made from foam due to the ease with which it could be cut to shape and hot glued together. Once we had made it, we then test fit the shroud, and did further tests to the line sensor and found it to work satisfactorily and so we taped the shroud onto the headboard, making sure not to allow any light leakage.

4.2 PCB/ amplifier

With the design of the line sensor finalised, the signal then must be sent to the Arduino in a range that the Arduino read. The Arduino can only read voltages ranging from 0-5V, and we would ideally have a single input to represent the offset of the line to the sensor. A Wheatstone bridge would be required to compare the two LDRs in one signal, though this value would be centred at 0V ± roughly 2.5V. This wouldn’t be usable on the Arduino, so an OP-amp is required to centre the voltage at 2.5V.

We opted to design a dedicated PCB that would be more robust than the breadboards we had been working with, and customisable to the dimension limitations we would have with our chassis. Rather than send 2 in/out leads to each LDR, we recognised that we could use the power and ground provided for the LEDs to be the ground half of the Wheatstone bridge, saving two wires from being sent to the sensor head adding flexibility. The positive half of the Wheatstone bridge was then implemented to the board amongst the OP-amp circuitry. A differential amplifier was used such that the differential of the Wheatstone could be kept but amplified and then raised to usable readings. The values were based off the circuit given in the lecture slides, with resistors of 33kΩ. These we’re selected because our Wheatstone bridge was only outputting roughly ±0.75V, which would be too small dynamic range to get accurate readings on the Arduino. Given that the resistors were not changed from 100kΩ, this would give us a gain of 3.03. This with the 2.5V offset gave us an output of 2.5v ±2.25V.

The circuit was built in Eagle using a 1.27mm grid to match the standard spacing of a breadboard. This was selected due to items such as capacitors being hard to locate the particular dimensions, but all fit standard breadboards. As such the capacitors we’re then drawn using standard PCB holes spaced out according to the components breadboard spacing, which proved very successful.

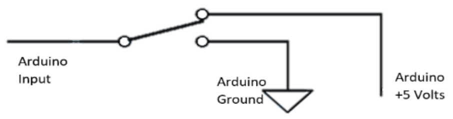


At 66mm by 71mm, the board was sufficiently compact with a large mounting surface to either avoid mistakes whilst drilling, or to provide a large empty surface which could be taped and hot glued which we ended up using. Without the limitations of columned breadboards, our PCB could be routed without the need for jumping over traces. Care was also taken so that all wires for the Sensor and the Arduino were closely positioned so that they could be grouped earlier (as shown above with the Green and Red boxes). This made the wiring much easier to follow and diagnose problems as the boards were clear of loose and single wires.

4.3 Motor Circuit

In order to operate the drive motors using the Arduino outputs, we made use of the h bridge board provided to us. This allowed a separate 6v battery to power the motors, which are then controlled through the PWM outputs of the Arduino. This means that we can still run the motors at full speed if desired, whilst not overloading the Arduinos output by attempting to draw too much current. This setup also allows us to have the ability to vary the motor speed through the use of the pulse width output's duty ratio. In order to drive the arm motor we used an additional h bridge board from the lab. Despite us not requiring a full bridge, and rather just a single transistor for the operation as the only control necessary was on/off we decided to use the additional board due to the simplicity with which it could be implemented into our design. The fact that there were two bridges per board also allowed us to have an additional motor, which was our original plan, however due to the limitation that we could only use one conventional motor and one servo motor the second bridge on the board was unused.

4.4 Arm Electronics



For the letter ejection a standard motor is used to drive a ribbon, connected to the Arduino with the use of ah bridge as discussed in 4.4. In order to move the arm assembly a servo motor is used, mounted such that its spindle acts as one of the arm pivots. This is possible as the torque the servo can provide being adequate to move the arm without any further gearing required. The servo is connected with two leads going to the main +6v and ground buses from the 6v battery, with a third pin connected to a PWM output of the Arduino. In this case an h bridge is not required as the Arduino output is not powering the motor, but rather setting the angle at which the servo operates.

The switch used for the activation of the Arm electronic and to stop the drive motors was a momentary SPDT, connected as shown such that the Arduino input is held high until the switch is pressed, at which point the pin is sent low. This allows us to know when the switch is pressed, whilst also avoiding having a floating input at any point.

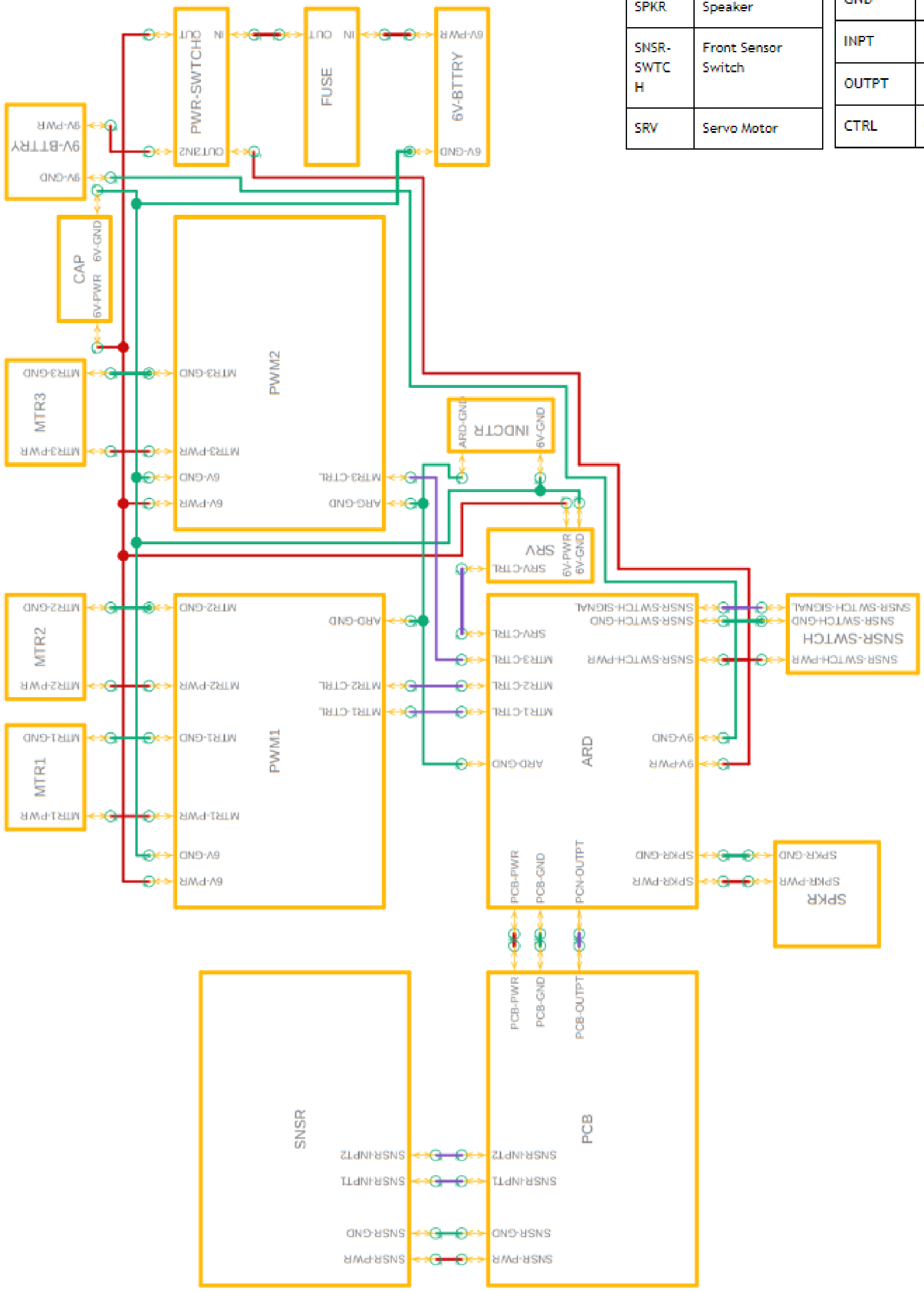
4.5 Wiring map

The wiring of the robot can be summed up to the assembly of different component blocks that are wired together. The above diagram is an ideal schematic of how each component is wired together. Despite this, the diagram gives a good representation of our actual wiring with the shared grounds for example on the PWMs and how the positioning of the choke inductor joining the 6v battery and the Arduino ground for the servo motor operation are accurate. The only inaccuracy to our actual wiring is that we did not use a bus system for the 6V battery positive and negative, though these all joined after the terminal capacitor and such would have no difference to the operation of the robot.

On the diagram you can see that the 6V battery's positive terminal is isolated from the circuit by a fuse and then a power switch. This was done to protect the circuitry in the case of a major electrical short, and in good practice to isolate the power switch to protect the user though the voltages are too low for this to be a real issue. By utilising a 6 pin switch, we were also able to power the Arduino from a 9V battery separately via the switch which means the 9V battery does not require being physically unplugged to stop powering the Arduino. This also has the added benefit of resetting the Arduino by turning it on and off as the code calibrates itself to the line every time it is powered up.

SNSR	Sensor
PCB	OP-Amp/Wheatstone Bridge PCB
ARD	Arduino
PWM	Pulse Width Modulator Motor Controller
MTR	Motor
SPKR	Speaker
SNSR-SWTCH	Front Sensor Switch
SRV	Servo Motor

INDCTR	Choke Inductor
CAP	Capacitor
PWR-SWTCH	Power Switch
FUSE	15A Fuse
BTTRY	Battery
PWR	Power
GND	Ground
INPT	Input
OUTPT	Output
CTRL	Control

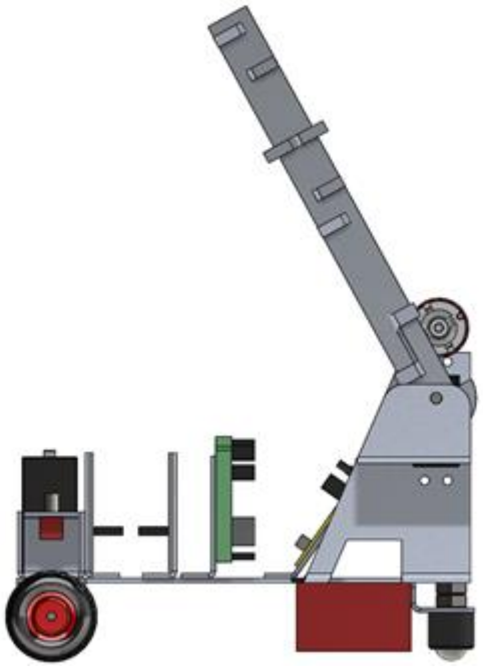


5 Mechanics

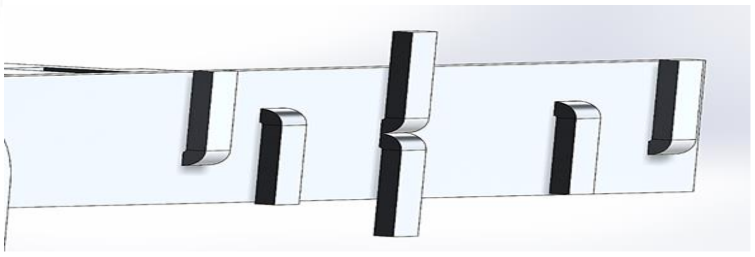
5.1 Arm Mechanism

Our arm mechanism, like the chassis, has been constructed from 2mm sheet-aluminium. The arm mechanism employs a drawbridge style movement, where the large black servomotor is connected directly to one side and controls the position of the arm as it rotates (see 4.4 above).

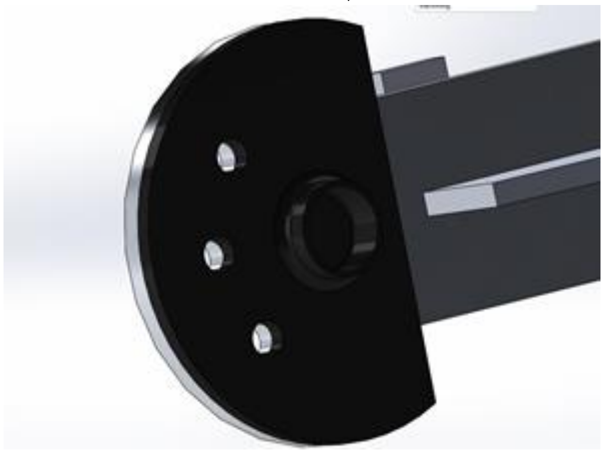
This mechanism has several strengths; it is both mechanically and geometrically far simpler than many ideas we had considered in our meetings, such as a scissor-lift-style mechanism, so the potential for error was reduced. Furthermore, due to the draw-bridge movement, the arm could theoretically have been infinitely long and not have exceed the robot's maximum top-down dimensions, bringing the envelope closer to the post box slot. The arm was held at a 'backwards' angle while the robot was in motion, bringing the centre-of-mass closer to the rear, and stabilising the robot.

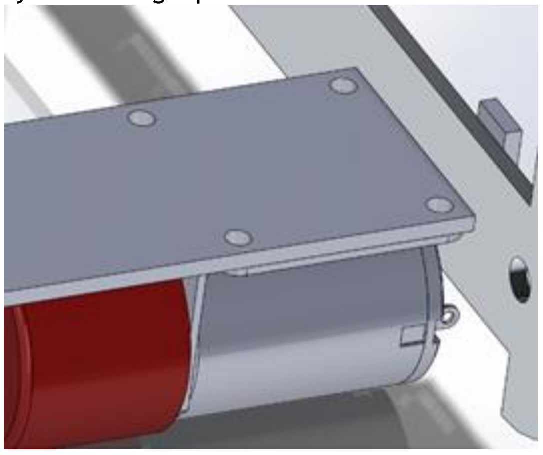


The arm has been designed to exclusively use mechanical fixings (as oppose to spot-welds or chemical adhesives), notably in the letter-housing area. This area has been constructed from 4 distinct layers, where tabs pass through the sides and protrude outwards. This not only simplified the assembly process, as the tabs were bent parallel to the sides by-hand, but also incorporated an element of flexibility, should one of the component parts have been faulty (in this case the part could have easily been replaced). Furthermore, this eliminated the possibility of the letter getting stuck due to it being too tight, as the housing geometries could be accurately controlled.



The arm has been secured to the servo motor with 3 nut-and-bolt fixings, as has the ribbon motor. While the parts for both the arm and the chassis have been formed using the water-jet cutter, the holes for these fixings could not have been formed with the cutter, due to the water stream being 3mm in diameter. Therefore, each hole has been cut by-hand using a pillar drill.





Once the robot arrives at the wall, a switch activates the arm mechanism, causing the large servo motor to rotate, lowering the arm. The angle which the arm rotates through has been carefully refined through trial-and-error, so that the letter points 'upwards' into the letterbox; as the letter exits the housing, gravity will attempt to force the letter immediately downwards, causing the front side to tilt. If the arm points upwards slightly, then the letter will require more time to tilt downwards, offering an amount of leeway while the letter is posted.

The letter itself is posted by a ribbon, which is looped around the letter in a ‘u-shape’, and then runs back along itself to the mottor. The ribbon is wound around the shaft of the motor on the arm; as the shaft rotates, the ribbon is pulled tight, pulling the letter out of the housing. We feel that this mechanism is particularly controllable (when compared to other potential posting methods, such as propelling the letter at speed), as the motor shaft can be rotated for a set period (see section 4.5). This period was found through trial-and-error. This ribbon pulley mechanism worked extremely well.

5.2 Issues with Arm Mechanism

While being constructed, the arm encountered several issues. Firstly, upon attaching the ribbon motor support bar to the arm, we noticed that the bar collided with the servo motor that rotates the arm, making it inoperable. We resolved this issue by simply cutting out a slot in the support bar which allowed it to pass around the servomotor.

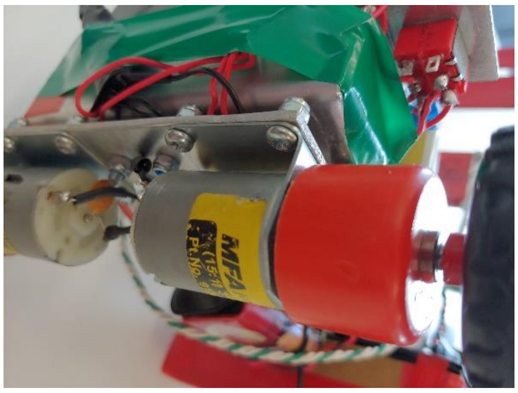
Furthermore, on the test day, an issue arose whereby the arm jolted uncontrollably after being lowered. This was due to there being a lack of damping around the arm’s shaft at the bracket (our previous bracket was not as accurate geometrically, so provided resistance to the arm’s rotation). We overcame this issue by fixing a number of bolts and washers to the shaft at the bracket. This worked very effectively in damping the oscillations.

The ribbon motor’s power supply is controlled by the Arduino; a trial-and-error process allowed us to dial in the time that the motor was powered. However, a lack of guides for the ribbon meant that it could freely twist while being reeled in. This issue wasted valuable development time prior to the test day and could easily have been averted by incorporating several guides to the ribbon.

5.3 Chassis and Electronics Housing

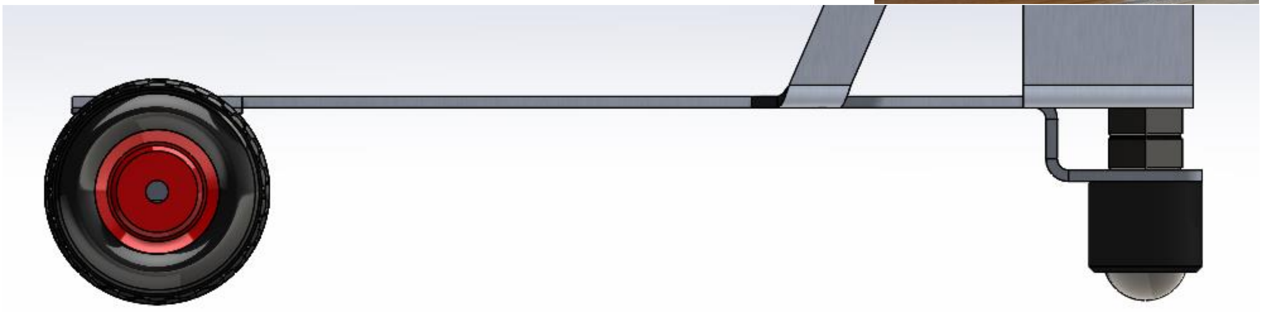
To design the chassis of the robot, we first looked at the restrictions to our design as detailed in the lab notes. Our biggest limitation was the footprint dimensions. The robot, ignoring wheels and probes, but including the letter and posting mechanism needed to have a maximum footprint of 140m by 200mm. This was an essential consideration during the design process. The arm mechanism, due to the size of the letter would not need to exceed this width limit, and because of our posting mechanism, it could be angled vertically, so could be however long we wanted. The design of this is as discussed in section 5.1 above.

We decided to make our robot structure out of 2mm aluminium sheet. We agreed that the 1mm steel sheet could be too thin and may bend, and that the 3mm acrylic could be too weak and brittle, making it tricky to work with and likely to crack. 2mm aluminium would be heavier than both options, but we decided that it would be more suitable as it would be quite easy to work with and still be strong. Having come up with these initial design considerations and restrictions, we began to design the chassis.



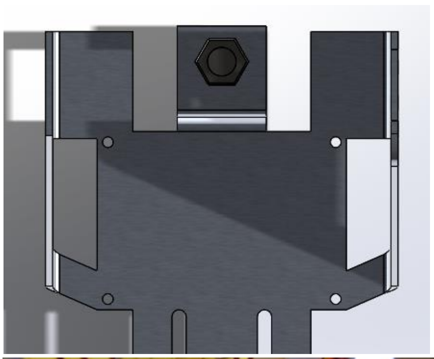
We wanted the robot to be as stable as possible, so wanted the powered wheels to be as wide apart as possible, with the omnidirectional wheel as far forward as possible. This then gave us a base plate of the chassis design, with the two powered wheels at the back, 90mm apart from each other, and the omnidirectional wheel at the front, 170mm in front of the powered wheels. We would bolt the motors onto the chassis using four 3mm bolts, as the holes are already on the motor stand for this, shown to the left. We were slightly concerned that the stand would bend under the weight of the robot, which it did, however not to an extent that caused any issues.

The omnidirectional wheel would be attached directly using the thread, through an 8mm hole, which we would punch into the aluminium. As the wheels are not the same height, we designed a flap to fold down for the front omnidirectional wheel, lowering the omnidirectional wheel by 10mm. This meant that on the flat pattern of the chassis plate, the omnidirectional wheel hole is 200mm from the powered wheels.

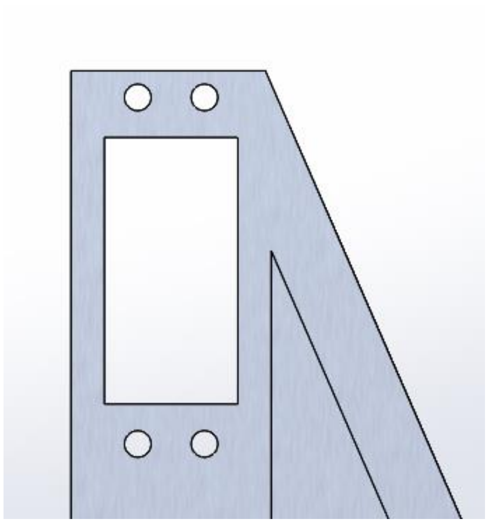


This kept the main chassis plate perfectly horizontal, which we needed for the line sensing module. It would also make it easier to work out the floorplan of the robot, and the height of the arm mechanism, which we had worked out previously - we had calculated the length of the arm mechanism which would be needed to get the letter to post into the letter box, as mentioned in section 5.1 above.

To reduce on weight, we removed as much material in between these wheels, resulting in a T-Shape. We decided we would then add to this basic spine as required. We knew that we would want the line sensing module as far forwards as possible, as this would give the motors more time to react to changes in direction. To do this, we added a plate behind the front omnidirectional wheel the size of the small breadboard (70mm by 50mm), as can be seen to the right.

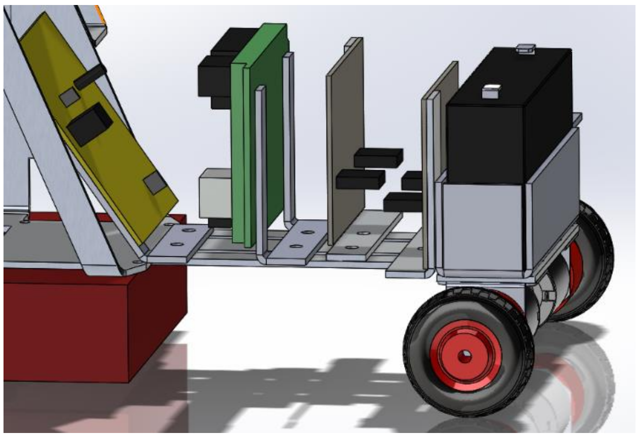


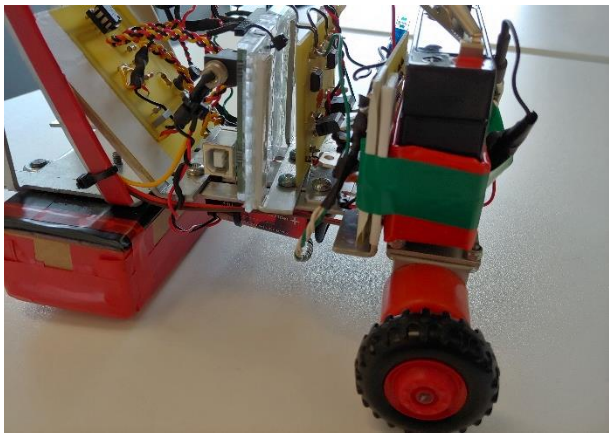
The next consideration was the mounting of the arm mechanism. In our calculations, we had worked out that the pivot point of the arm should be 100mm off the ground to best post the letter into the slot. We had decided to power the arm mechanism using 2 of the large black servomotors, a decision which will be discussed later. This meant that we effectively needed a second level of the robot. To do this, we created two arms or “wings”, which would then be folded up. Each wing had a slot the size of the servomotor (40mm by 18mm), along with the four 4mm holes to bolt the motor to the aluminium. These motors would fix directly onto the arm mechanism, using the full circle black HDPE attachment plates provided with the servomotor, attached to the arm using 3mm bolts, seen below (next page).



The picture to the left is a screenshot of the servomotor slot in the Solidworks drawing, and the picture on the right is the actual servomotor attachment to the arm mechanism. The slot worked very well, but the attachment itself was not as secure as we would have liked but did the job effectively.

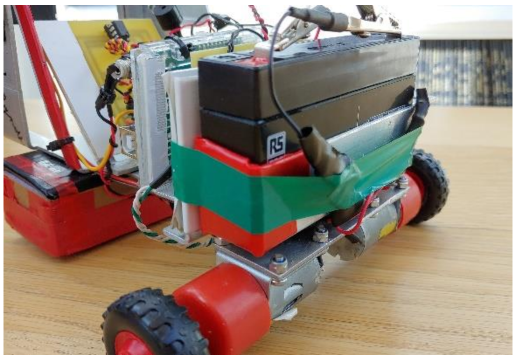
Our next design was surrounding the electronics. We knew that we required two breadboards, a 6V battery, and Arduino to be mounted on the chassis, as well as the line sensing module. The issue here was that we did not know the exact dimensions of any of these boards, including the line sensing module until we had completed that section of the electronics. To overcome this issue, we developed a modular style mounting mechanism to mount all the electronics onto the chassis, but still allow for movement. More importantly, we made sure that the design would allow the electronics to be worked on when the robot was completely assembled, and that the Arduino port would be accessible so we could upload code to it while on the robot. This “modular system” was to have two slots running along the length of the robot. The electronics boards would then be attached to tabs or brackets which would bolt onto the slots. This would allow the boards to be replaced and be moved around on the robot itself, so if any of the electronics changed from what we had planned it could be adjusted to facilitate the change.



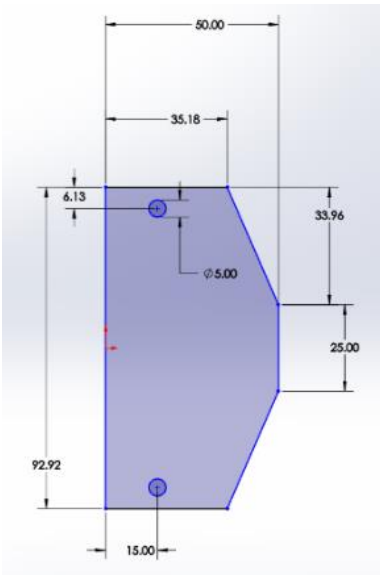


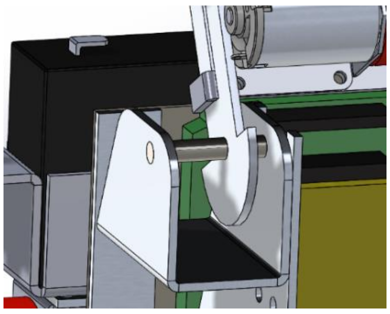
A similar approach to the line sensing module was considered, however we were sure that this would not change, so designed the chassis plate to fit the specific breadboard size of the line sensing module. Despite our initial belief, this breadboard was changed for the final robot design, but we managed to attach it very nicely despite this, using a tap and die to thread the holes for the bolts in the line sensing module.

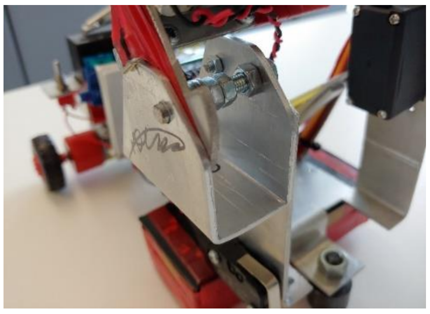
As well as brackets for the breadboards and Arduino, we made a casing for the battery, which included a slot for a switch, allowing us to turn the robot on and off when necessary. The battery was designed to attach just above the wheels, as it was the heaviest part of the electronics, and this was where it would get the most structural support. This can be seen in the previous Solidworks drawing, and the picture to the right.



While it was not in the original design, to attach the arm mechanism, we made a U bracket after we had done all the waterjet cutting and finished everything else on the chassis apart from the arm mechanism. This meant that we could make sure that the centre of the hole on the U bracket would be at the same height as pivot point on the servomotor, which would ensure that the arm mechanism was horizontal and flat. Our U bracket is slightly different in the SOLIDWORKS, due to the slight differences in the bends on the wings of the main chassis. Also, we added washers to our actual robot bolt to ensure that the arm was held as securely as possible, as shown in the pictures below.







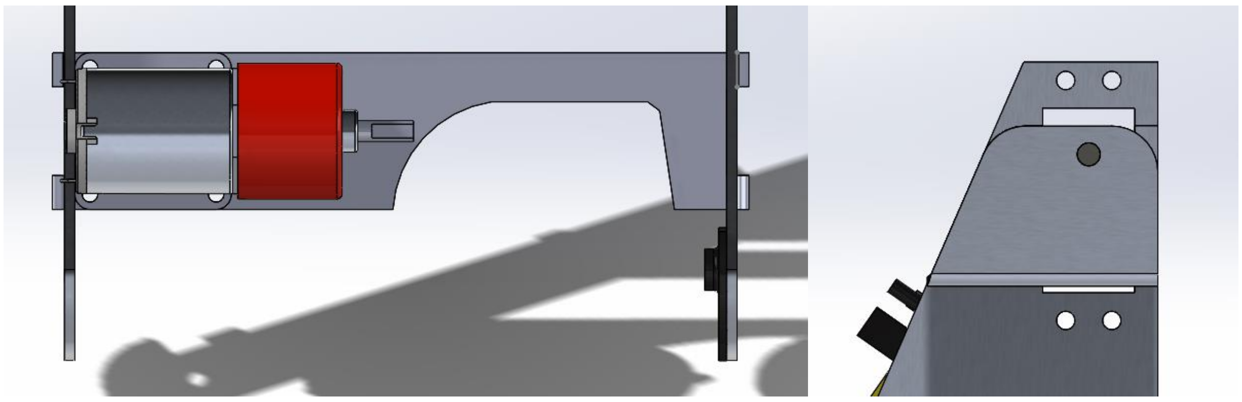
All except for two of the parts (battery case and U-bracket for arm mechanism) for the robot were cut out using the water jet cutter, which meant that we had to design the entirety of the robot before most of the electronics was built, hence the modular style of design. The huge advantage to this was the accuracy of the robot. We knew that the holes in the electronics board tabs would line up perfectly with the slots on the robot chassis, and that the holes would line up perfectly for the motors and arm mechanism.

5.4 Issues with Mechanical design

The major issue which we had with the manufacture of the chassis was issues with the waterjet cutter. Ours was cut out 2 weeks later than every other groups. This meant that further down the line, when the electronics team needed the chassis to work on the line sensing module, the mechanics team also needed it to work on, so we had to divide our work very carefully to ensure that both the mechanics and electrics teams could progress each lab session without hindering the progress of the other half of the team.

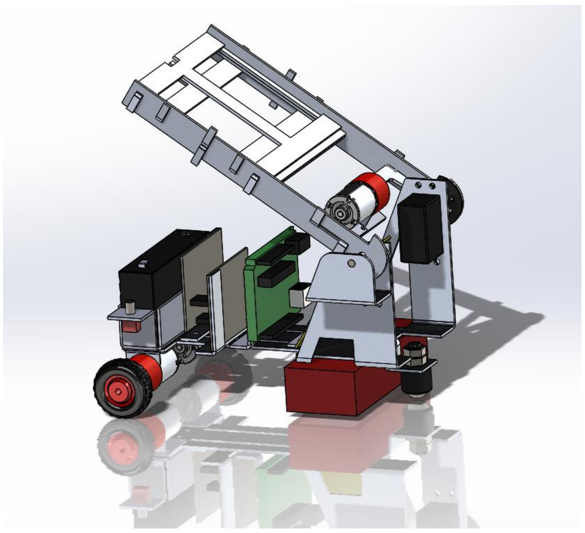
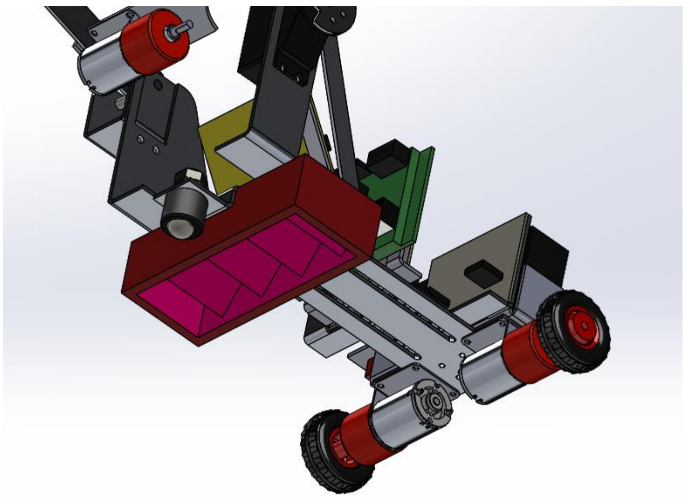
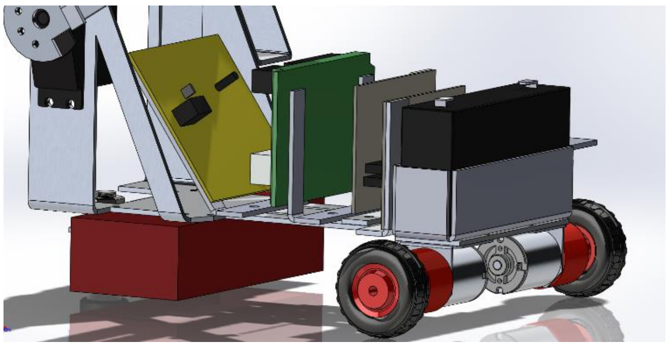
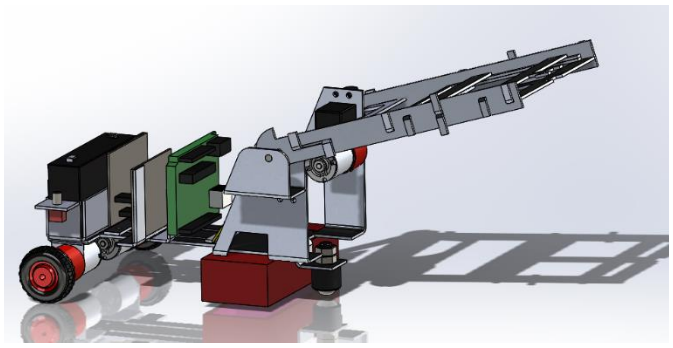
The main problems with the design were due to the arm mechanism, and the way in which we designed the two parts (arm mechanism and chassis) almost completely separately. As mentioned in section 5.3 above, we had designed the posting mechanism arm to be powered and connected to the chassis using two large servo motors. However, when we asked for a second motor we were told that we were not allowed to use two. This was a huge mistake, as we should have asked whether we would be allowed to use two motors before we had designed the chassis to connect to the arm mechanism purely using the two motors. Instead we built the U-shaped bracket as describe in section 5.3 above, and then put a bolt through the side, holding the arm mechanism in place. This was not a particularly elegant solution, but it worked effectively.

When we came to attach the arm mechanism to the robot, we realised that the plate on the arm mechanism which the ribbon motor sits on hit the top of the “wings of the robot”. We had attached the arm mechanism before, but not with the ribbon motor and its plate also attached to the arm. To overcome this problem, we cut a slot in the plate on the half that didn’t have the motor on it (this part had to go around the servo motor, and then cut down the height of the wing on the side of the plate that housed the ribbon motor, which can most easily be seen in the SOLIDWORKS.



This issue was purely down to a lack of testing, as if we had made cardboard models of the chassis and arm, we would have foreseen and could have avoided this problem. Our solution worked, but if we had had more time, we could have come up with a better solution for this problem, designing the army to attach directly onto the chassis wing. The advantage of the U-bracket in this respect was that it kept the bolt horizontal and fixed in position, so was a more secure attachment to the arm.

5.5 SOLIDWORKS Drawings

6 Testing

6.1 Evaluation of Mechanical Performance

On the Test Day, the mechanical parts worked well, despite a few issues early in the day. The arm was not held securely enough by the U bracket to begin with, which caused the servo motor to oscillate very quickly up and down, resulting in the arm bouncing up and down. After diagnosing the problem, and seeing how to fix it, we added some washers to the bolt on the arm, and effectively clamped the arm to the side wing of the robot. This worked perfectly, and the rest of the mechanics went smoothly throughout the day.

6.2 Evaluation of Electronic Performance

On the final testing the electronics performed as expected with no electrical faults. Having had the electronics assembled a few days prior, we had already had time to check everything had been put together without breaking any connections or accidentally shorting out on the chassis. The sensor module and OP-Amp circuit successfully sent a stable signal to the Arduino within in the bounds of the Arduino analogue input pins, allowing the code to perform as optimally as possible.

6.3 Evaluation of Coding

On the final test the code performed well, completing all tasks successfully, with no bugs or faults occurring. Having done practice runs before, we can confidently say the final test was not a fluke. The robot completed the same multiple times successfully. This shows a robust and well-structured code. A problem occurred when we started trying to speed up the motors. This was not due to a bug in the code and was more to do with the control method. With more time I think we could have run the robot faster while still successfully following the line. Overall the code worked exceptionally and, given more time, the robot would have been able to complete the task in a shorter time period.

6.4 Overall Performance

In testing, our robot completed the test, following the line and posting the letter perfectly on its first attempt. The robot performed as expected, with all aspects behaving as intended. Though we believe with more time the robot could have followed the line faster, and weighed less, we are happy with our 5th fastest and 4th lightest accolades.

7 Conclusion

7.1 Areas for Improvement

If we were to attempt the project again, we would have changed a few things on the robot, and would also have conducted the project differently. We believe that acrylic would have been a more suitable material to use for the robot. While the aluminium was stronger and easier to work with, the robot didn't require this additional strength, and could have used the weaker acrylic just as well. This would have enormously reduced our overall weight and increased the speed as a result. We would have changed the design for the arm mechanism connection to the robot itself, but this issue was due to not checking whether we were allowed a second servo motor, rather than a design flaw. Another improvement would have been to add a second line sensing module behind the first. This would have smoothed the run of the robot significantly, as the it would have been able to react to even slighter changes in the line. The ribbon envelope ejection mechanism worked very well, but using the large motor on it seemed unreasonable, and we would have liked to either have the motor mounted on the robot chassis or used the small servo motor to control this. Moving the motor would have reduced the weight and torque on the servomotor, making the entire arm mechanism more reliable. As already mentioned, the code could also have been improved to make the robot faster.

As a team we worked well together, however we could have improved on the way in which the electronics and mechanical teams worked together. Having a clearer picture of the final electronics boards would have helped the mechanical design and making the chassis earlier would have helped the electronics team, although this issue was out of the team's control.

7.2 Overall Conclusion

Overall, the group worked well together, delivering a successful robot and completing the challenge, despite the issues that arose throughout the duration of the project.

For example, the electrical team managed to overcome the complications that arose during assembly process, namely a faulty op-amp which needed replacing and caused some lab time to be wasted whilst the issue was diagnosed, and the issues with circuit boards discussed earlier in the report.

The test day proved to be challenging for all 3 teams, presenting hindrances that had not occurred the previous day, such as a consistent jolting in the arm with the revised bracket. Our first run was completely successful, demonstrating that the robot can follow the line, and that the posting method works. Despite this we did two more test runs using code which would allow the robot to follow the line more quickly, but which reduced the reliability of the line following system. Attempting the 2nd run with the updated code resulted in a failure to follow the line; whilst the 3rd run saw the robot arrive at the post-box successfully, however we had failed to load the envelope into the housing, deeming the run unsuccessful. This run was actually faster than the first by around 2 seconds and so had the envelope been loaded it is very likely to have been our best run, as the posting mechanism also appeared to operate correctly. Fortunately this improved time would not have impacted our speed ranking meaning that whilst annoying, the mistake did not cause any grading differences.

In conclusion, the group thoroughly enjoyed participating in the robot project, gaining knowledge in problem-solving, team coordination, and in the design-build-test procedure.