Mechatronics - Technical Report
EE30220

Integrated Design Engineering - University of Bath

Sam Riddell-Webster
1 December 2019

Contents

1 Summary

The Following paper outlines the method implemented for an integrated Mechatronic system. The aim of this system was to locate magnets within an XY Gantry with the use of Hall effect sensors and mark them using plastic cups. What follows is an in depth description of the mechanical mechanism and the Software method which was implemented. Both aspects were then discussed in detail and analysed against existing practice to consider the quality of the methods that were used.

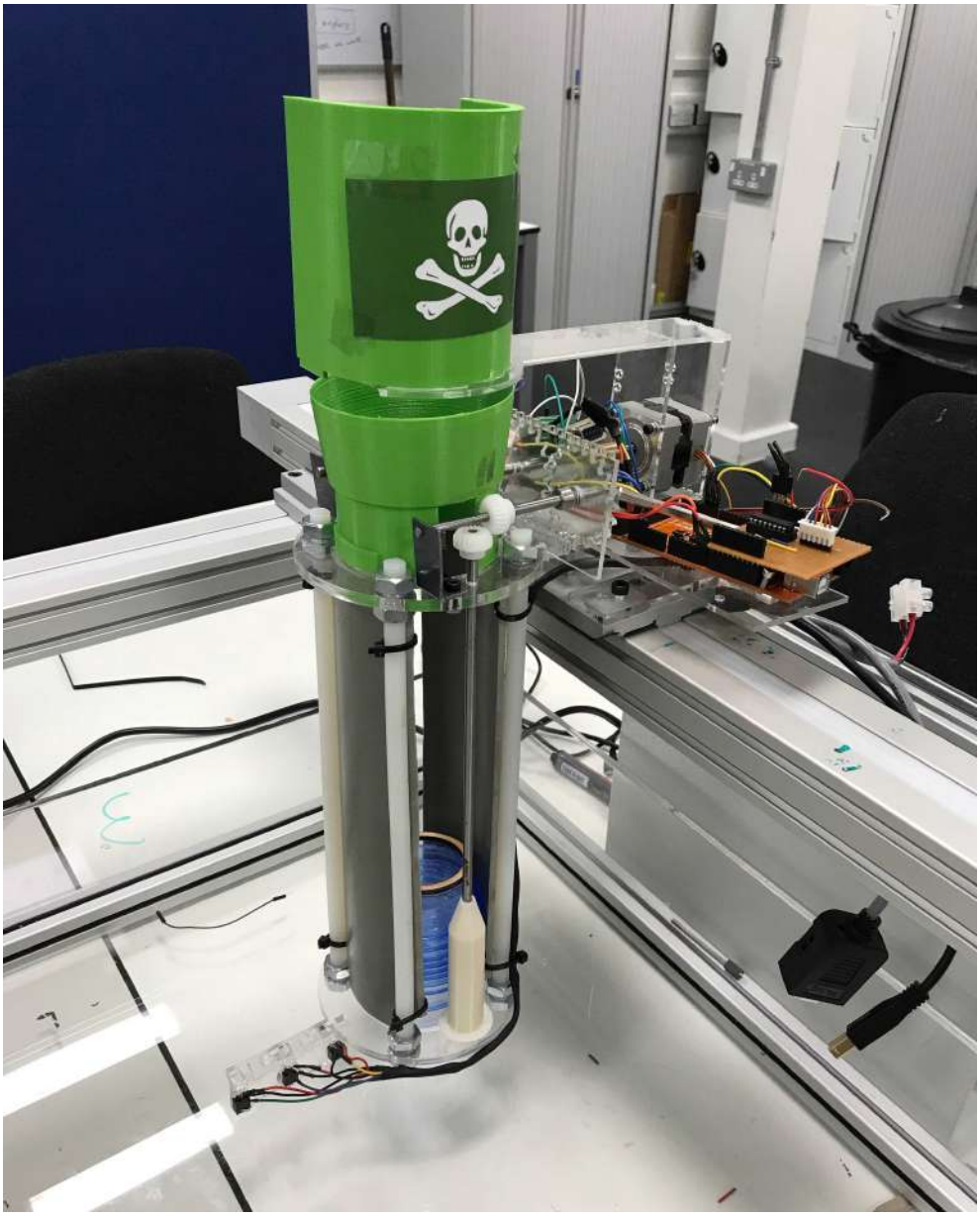

Figure 1: Final System

2 Introduction

2.1 The Task

The task was based around the use of a $500mm^2$ XY Gantry and bed. Magnets of two sizes were placed on the bed. The task was to build a mechanical system with integrated sensor head, Electrical hardware and software for control. The sensor head could use up to three Hall effect sensors for locating the magnets. The mechanical system would have to collect plastic cups and release them at the correct location. This Mechanism was to be integrated with the electrical hardware and software capable of producing a fully autonomous system.

2.2 Design Specification

		Specification Point	Essential/ Additional	Source
Find Treasure	1.	Find treasure to within 35mm.	Essential	Brief
	2.	Find both small and large magnets.	Essential	Requirements
	3.	Scan and mark magnets in shortest time possible.	Essential	Brief
	4.	Scan along most efficient path.	Essential	Requirements
	5.	Cover largest area possible.	Essential	Requirements
	6.	Maximise scanning range and accuracy.	Additional	Requirements
Move to Treasure	7.	Move land ship using two stepper motors.	Essential	Brief
	8.	Control both stepper motors at the same time.	Additional	Brief
	9.	Use feedback to control stepper motor location.	Essential	Course Material
	10.	Move to treasure as fast as possible.	Essential	Requirements
	11.	Use the most efficient scanning path.	Essential	Requirements
	12.	Use PID control to move land ship smoothly.	Additional	Course Material
	13.	Safety features to prevent damage to landship.	Additional	Requirements
Mark Treasure	14.	Pick up markers from on/around gantry.	Essential	Requirements
	15.	Collect markers in a controlled motion.	Essential	Requirements
	16.	Locate up to 15 treasure locations.		Requirements
	17.	Securely hold markers whilst on landship.	Essential	Requirements
	18.	Gently release markers over magnet.	Essential	Requirements
	19.	Encapsulate magnet within the markers.	Essential	Brief
	20.	Avoid previously marked locations of the magnet.	Essential	Brief
	21.	Use plastic cups as markers	Essential	Brief
Land ship	22.	Load markers from gantry onto landship.	Essential	Brief
	23.	Locate all electronics and hardware on the landship.	Essential	Requirements
	24.	Keep landship weight to a minimum.	Essential	Requirements
	25.	Less than 14Nm moment over gantry	Essential	Datasheet
	26.	Less than 500N load on gantry.	Essential	Datasheet
	27.	Carry out simple tasks externally, electronically.	Additional	Requirements

Figure 2: Specification Point Sorting

2.3 Main Aims

The initial design stage and generation of design specification lead to two key goals, Primarily, the task must be completed fully. In this case all magnets located and marked fully within the cup area. Secondly, the task operation was to be completed within the quickest time frame possible. This was an important criteria as the nature of the task was a competition against other crews working on the same task. The key points of the design specification are highlighted below. Given the constrained time frame of this task and the reduce man power of the team, this project was under taken on the principles of Occam's Razor.

"The Simplest solution is almost always the best"

Essentials

- Locate magnets to within 35mm of the centre minus the magnet radius.
- Find the most efficient scanning and marking method.
- Control the release of markers over all magnets.
- Find all treasure location.

Additions

- Drive both gantry motors at the same time.
- Simplify the process wherever possible e.g. collect all cups so no need to solve traveling salesman problem.

Above are some of the core specification points selected from Figure 1.

2.4 Demonstration Outcome

The final test failed to achieve our primary goal, as a result the secondary goal of achieving a quick time was also failed. These failures were likely due to a failure in the wiring of the hardware which was vital for the location of the ship within the gantry. However, tests from both before and after presentation test demonstrated that the task could be completed in approximately 1 minute 10 seconds, marking all the magnets. From, this it can be concluded that the concepts behind the deign and the methodology used for initial cup collection followed by scanning and marking were sound.

3 Initial Design Stage

3.1 Field of Study Allocation

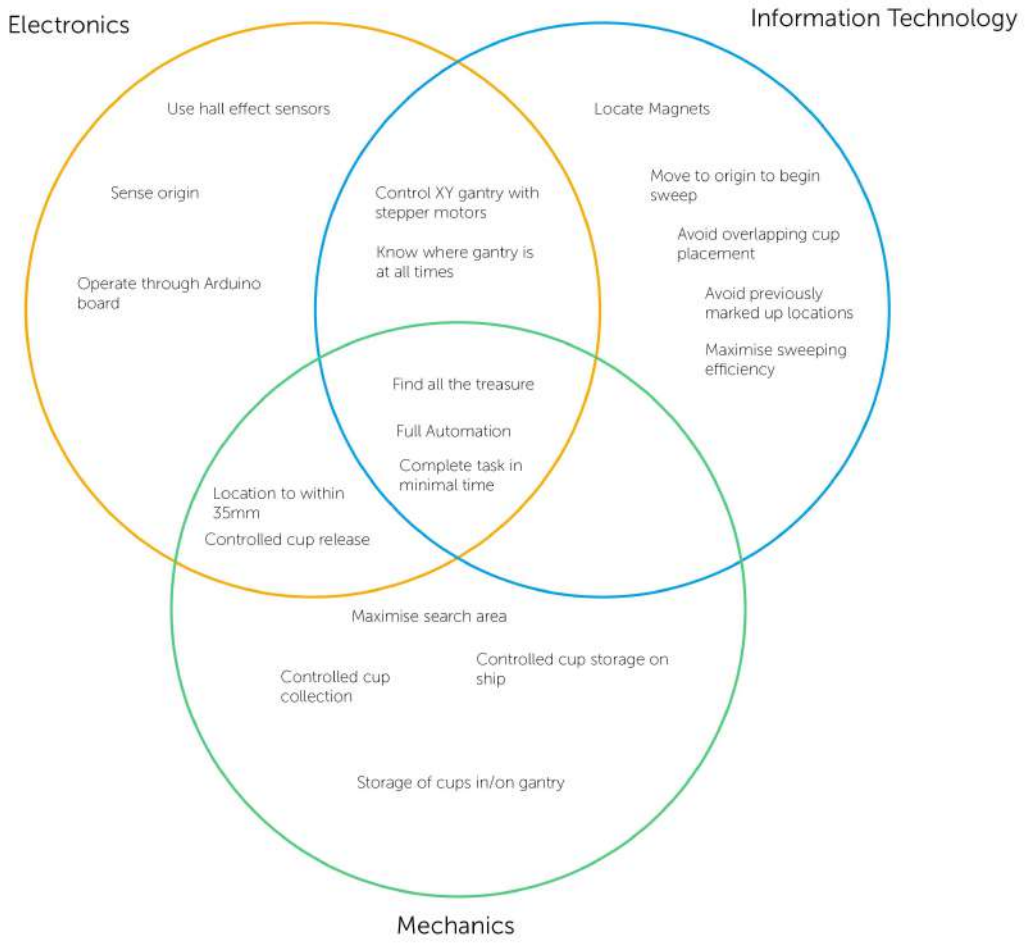

Figure 3: Specification Point Sorting

Figure 3 shows the three key areas within a mechatronic project, the essential criteria from the design specification have been sorted into these three categories. This was a useful tool in the early design stage, which aided in the process of task delegation. This report will focus on all points under the purview of information technology, as well as those mechanical points that link closely to software, such as the scanning system. The mechanical drivetrain will also be discussed. These represent areas where the author took a leading role.

3.2 Design Methodology

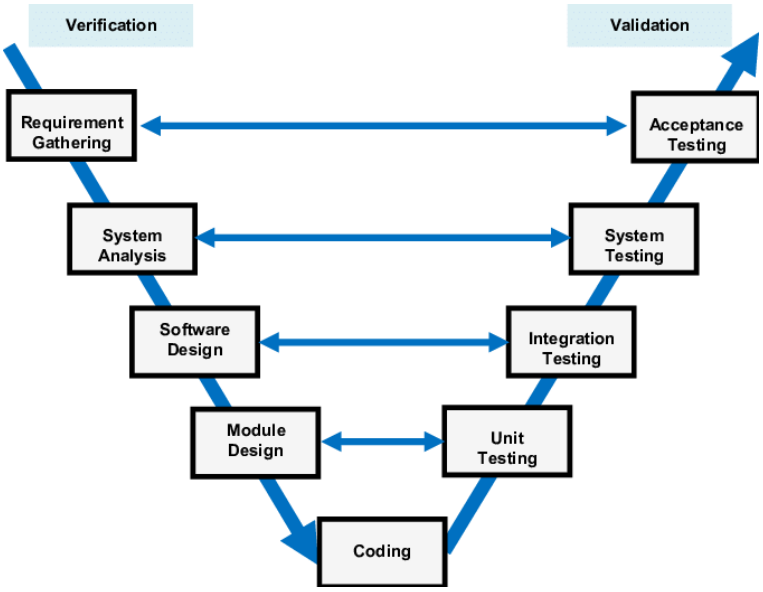

Figure 4: V Model of Design Process

The V model is a design process which uses Verification to outline the beginning of a project and Validation of the theories and principles based on testing. This model was used as a basis for this project. Constant testing and development used to back up initial design concepts.

3.3 Concept Generation

The initial design stage involved brainstorming of concepts for the two key mechanical functions:

- Cup Collection
- Cup Dropping

A major consideration during the design stage was the potential benefits of a system that could collect all cups initially, before completing a scan. The two most significant being:

- Reduce overall required travel of the gantry.
- Remove the requirement for complex position finding algorithms such as travelling salesman method.

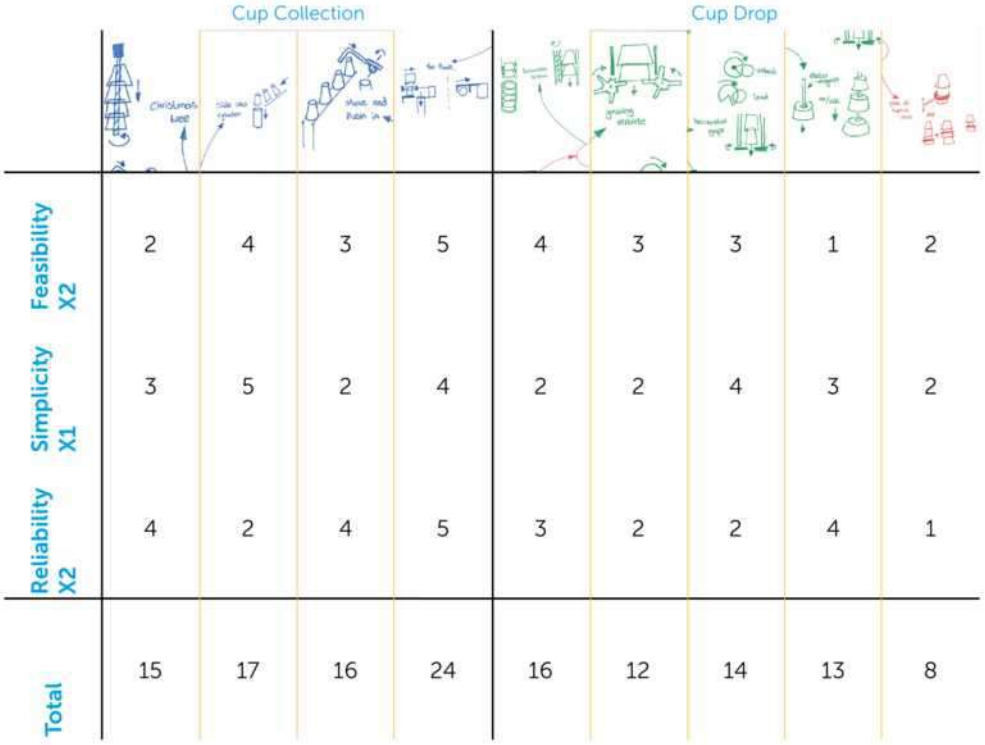

Figure 5: Concept Selection Chart

Figure 5 shows the scoring of potential methods based on a weighted consideration of three key features: Feasibility of implementation, Simplicity of Design & Concept Reliability. The outcome of this concept selection was a collection method where stacked cups are pushed off a plank by controlling the gantry. The selected cup dropping method was an external screw system which drops cups with every full rotation.

4 Mechanical Solution

4.1 Drivetrain

The mechanical drivetrain was designed based on the principle of simplification which was highlighted as a key Requirement. The primary success of this system was that it allowed two screws to be driven by a single stepper motor.

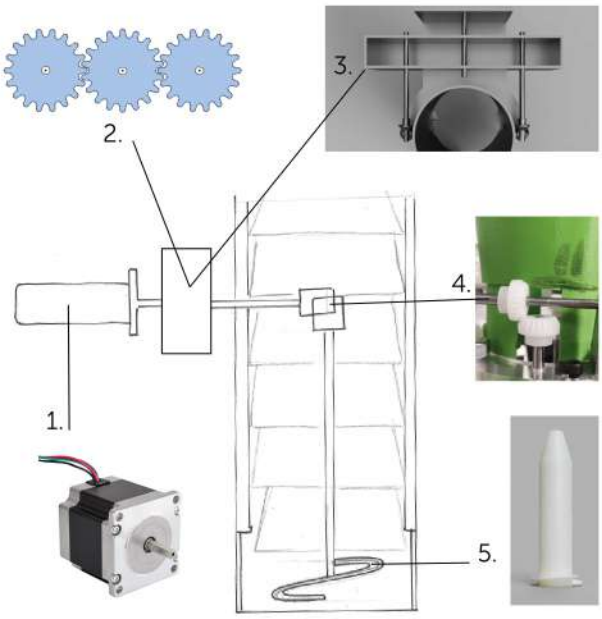

Figure 6: Drivetrain Subassembly Diagram

1. The mechanical system was driven by a Stepper motor, this allowed for accurate control or rotation. Important in this situation, as the rotation needed to be exactly 360° for each cup drop.

2. The gear box consisted of three 1:1 gears, with the central gear driving the outer two.

3. The gear set up allowed one gear to drive two outputs whilst simultaneously achieving the required separation. This separation was required to fit the shafts around the cup storage housing.

4. Two 90°bevel gear couplings were used to achieve the vertical drive. This allowed the system to be driven without housing the motor vertically above the drop point which would have interfered with the cup loading system.

5. Two external screws were used to drop one cup at a time from the housing. The use of two insured the cups were dropped evenly and didn't get skewed either in the chamber or after release.

4.2 Drivetrain Development

Initial designs for the external screw considered an extended screw that interlocked with all the stored cups for more accurate cup control within the storage chamber. However, this idea was simplified down to a single rotation of screw. This enabled the the cup loading system to be much simpler as each individual cup no longer needed to be loaded into the screw. Through further development using 3D printing, it was discovered that the top of the screw could pinch the cups, preventing release. Reducing the outer diameter of the screw at the top, with it progressively increasing down the spiral, lead to smoother and more reliable cup release. This is shown in Figure 7. With this addition, the cups were dropping well when aligned correctly but would often fall unevenly. To help constrain the cups by preventing lateral movement, the inner diameter of the spiral was increased, a tapper was the added to the top to encourage correct alignment.

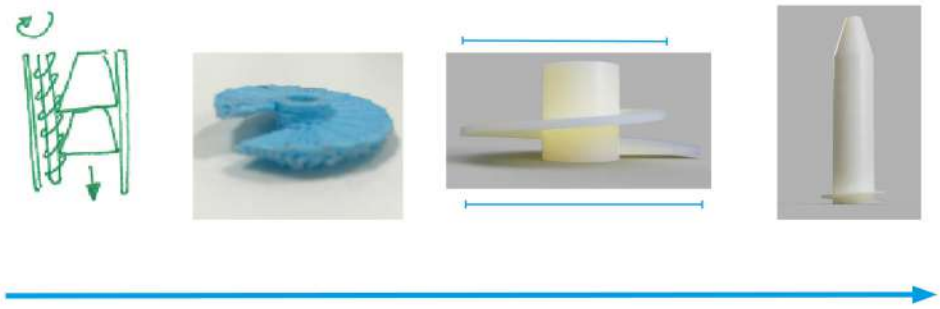

Figure 7: External Screw Development

4.3 Scanning System

4.3.1 Scanning System Limitations

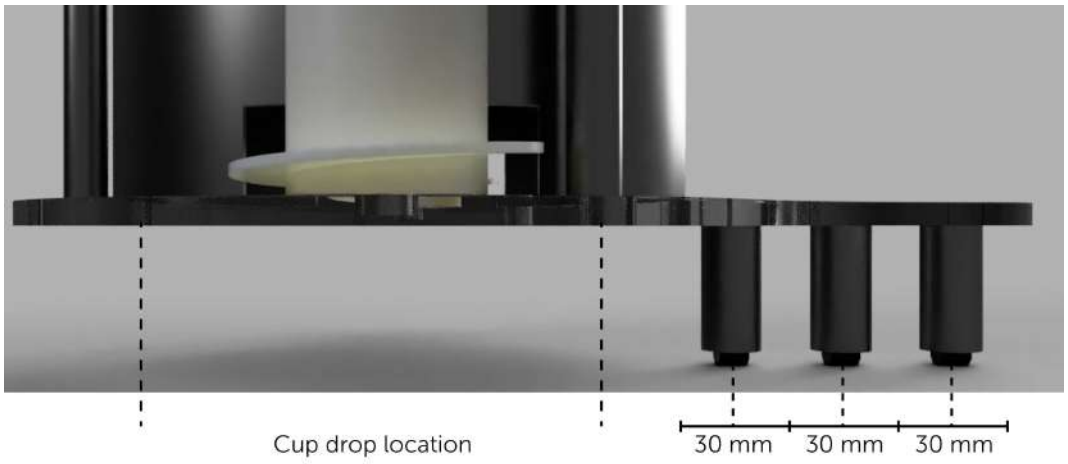

Figure 8: Sensor Head with dimensions

Figure 8 shows how the sensor head was orientated with the bottom of the cup housing unit. The sensor head was required to be lower than the bottom of the cup housing, the was due to two design limitations:

· The cups could not be altered to lower the 35mm.

· The sensor head could not be farther than 20mm from the bed.

This second point was essential in order to comply with specification points 2 & 6. As a result if the sensor head passed over any area were a cup could have been dropped then the sensors would dislodge the placed cup. This limitation of the design made it essential that the scanning path never moved backward or covered the same ground twice. For this reason a linear scanning method was adopted.

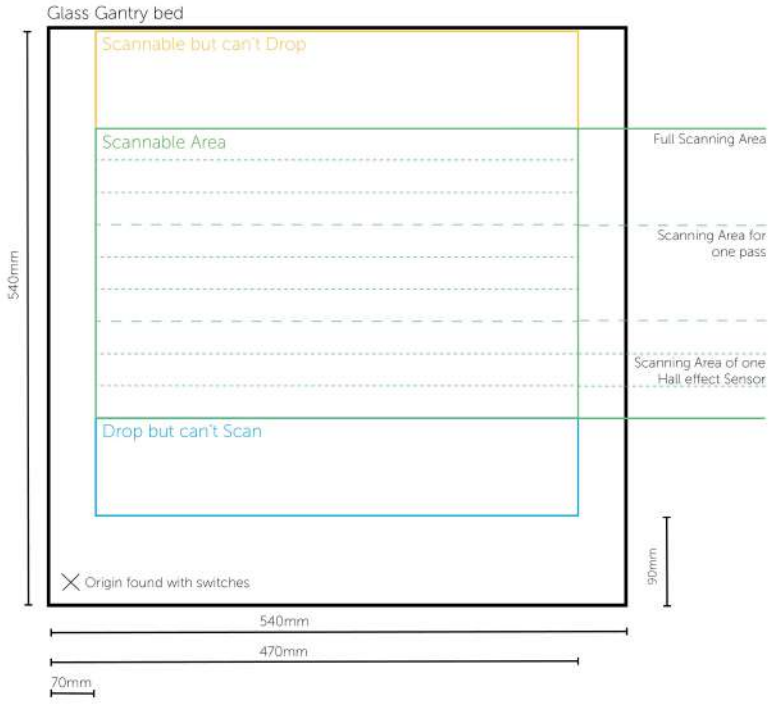

Figure 9: Gantry Bed Scanning Method

4.3.2 Scanning Method - Constraints on Software

To initiate the search, the system started by finding the Origin (simulated by two switches). From that point it could begin horizontal scans as seen in Figure 9. Each sensor had a separation of 30mm as shown in Figure 8. This allowed each Hall effect sensor to have a scan path width of 30mm. Using three sensors, the total scan path was 90mm. A significant draw back to this design was the reduced area that could be scanned and marked. The sensor head was located in-front of the cup housing. Therefore, the 90mm strip closest to the origin could be marked by a cup but couldn't be scanned. Similarly, The last 90mm strip could be scanned fully but could not be marked with a cup, as the sensor head would interfere with the wall of the gantry.

4.3.3 Scanning Method Discution

The sensor head set up and scanning method were selected as they fulfilled multiple specification points. Point 3, 4 & 6 all to the speed, efficiency and accuracy of the scanning.

- Speed: The integration of scanning and dropping decreased the required travel distance which increased the total speed of operation.
- Efficiency: By scanning the bed linearly, the vast majority of gantry movement was spent scanning, with only the movement between paths spent not scanning. With four scanning paths, 80% of movement was spent scanning.
- Accuracy: By moving in X and Y directions individually, the location method was simplified this made it easier to locate magnets based on PWM coordinates. This made accurate location possible, in the Y direction.

The linear setup achieve these requirements as well as simplifying the software requirements. It also removed the need to control both motors at the same time, which simplified the electrical requirements. As a result of this the two scanning strips highlighted in Figure 9 were lost. This had a negative effect on spec point 6, maximise scanning area. Future iteration could include a hinged sensor head that could be raised to drop a marker in the yellow scan strip in Figure 9.

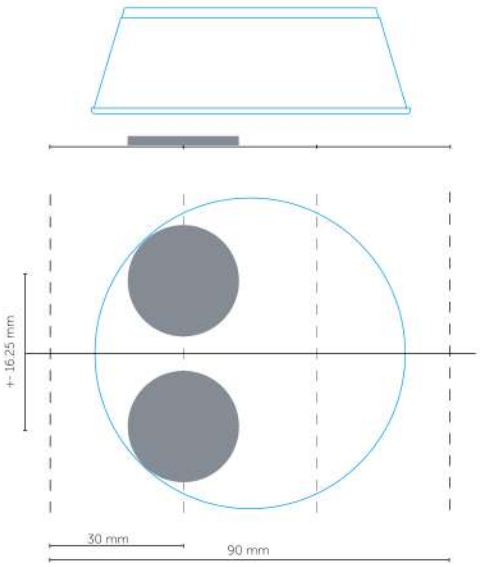

Figure 10: Cup Drop Position Accuracy

Figure 10 shows the required accuracy for the magnet location based on the physical limitations. Assuming the software is able to locate the magnet to the nearest of the three Hall effect scan paths. Plastic cups with an internal diameter of 70mm and the larger magnets with diameter of 25mm were considered. Based on these assumptions and dimensions, the software would need to be able to locate the magnet to within an accuracy of $\pm$16mm. If this is achieved then all magnets would be located fully within the cup, no matter where within the path they are orientated.

5 Software Solution

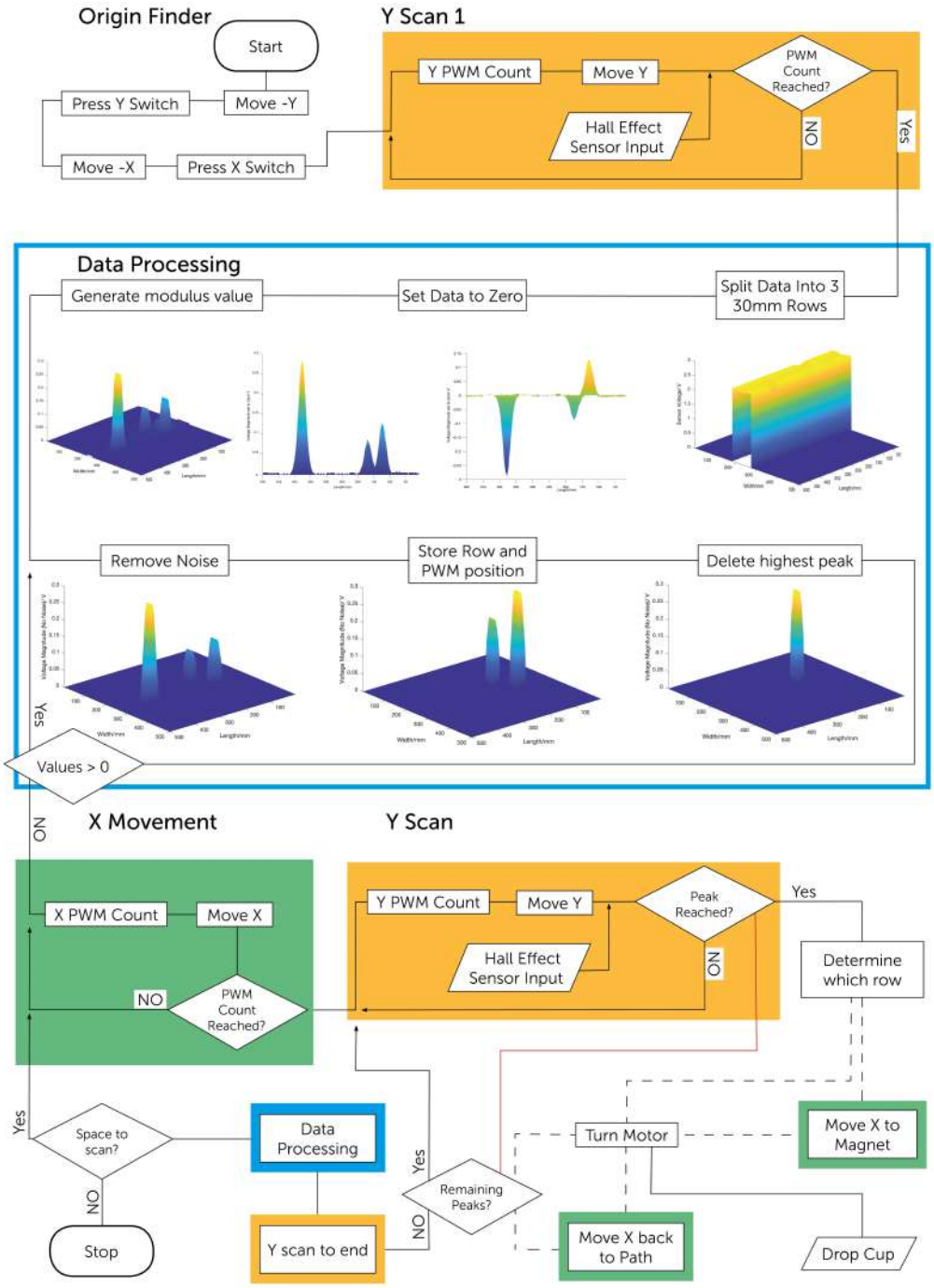

Figure 11: Software Flow Chart

Figure 11 shows the methodology behind the software. As mentioned in correlation with Figure 9, the system started by moving in a negative directions, first in the Y direction, then in the X. Stopping when the gantry arm came into contact with the hardware switch built into the side of the gantry. Scanning was then initiated along the first path. Whilst scanning, a pulse width modulation count was run on the hardware. When this reached a pre- determined value the gantry movement was stopped. The data collected by the Hall effect sensors was then processed to locate any magnets (see 5.1 for detail). Once Data processing was complete, the gantry would move 90mm in the X direction, again controlled by the PWM count. The gantry would then start the next sweep back toward the origin. If any magnets were located in the first sweep, the gantry would stop. It would then move 30mm in the X direction, to the row containing the magnet, if required. Then a cup would be dropped, by turning the motor 360°. The Gantry would then return 30m to the original path and continue scanning.

5.1 Data Processing

In order to control the gantry to move the point of a magnet, the data from the Hall effect sensors had to be processed into a co-ordinate.

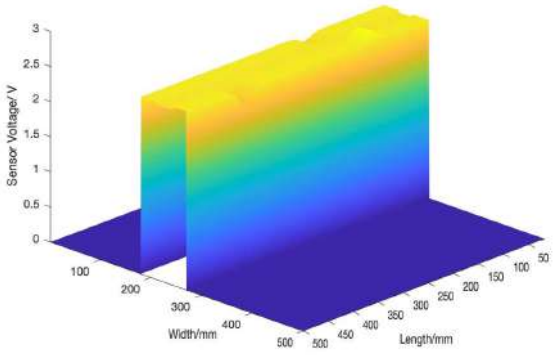

Figure 12: Raw Data collected by Hall Effect Sensors

Figure 12 shows the voltage from the three hall effect sensors as they scanned across the bed of the gantry. There are a few small fluctuations both negative and positive from the base value of 2.5V. These highlight that the sensor has passed over a magnets. The length represents the distance across the gantry bed, this was determined from the PWM count sourced from the hardware.

Figure 13 shows the data from Figure 12 with the default value set to zero. This was done by finding the modal average of the data and removing this from the initial value. This one done rather than simply minus 2.5 as it provides more accuracy and margin for error.
Lines 43- 49 find the modal average and remove from original data:

```
function(TheY1)
Line No.
43          avg1 = mode(Line1);        % sets to zero approximatly
44          avg2 = mode(Line2);
45          avg3 = mode(Line3);
```

```
47      Line1 = Line1 - avg1;     % sets to zero by removing average
48      Line2 = Line2 - avg2;
49      Line3 = Line3 - avg3;
```

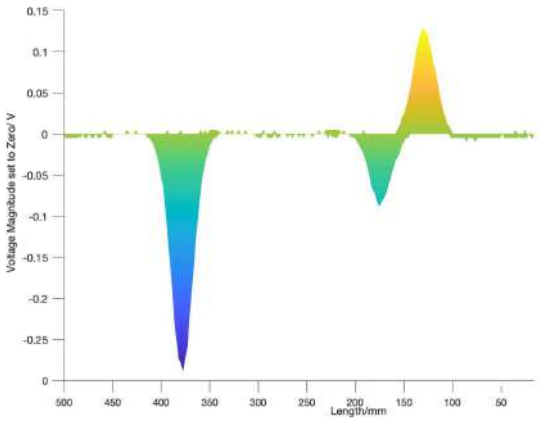

Figure 13: Voltage set to zero

Figure 14 shows the magnitude of the data from Figure 13. This gives all positive values which reduces the chance of errors. This was done using lines 51-53:

```
function(TheY1)
Line No.
51              Line1 = abs(Line1);          % Finds Magnitude of data
52              Line2 = abs(Line2);
53              Line3 = abs(Line3);
```

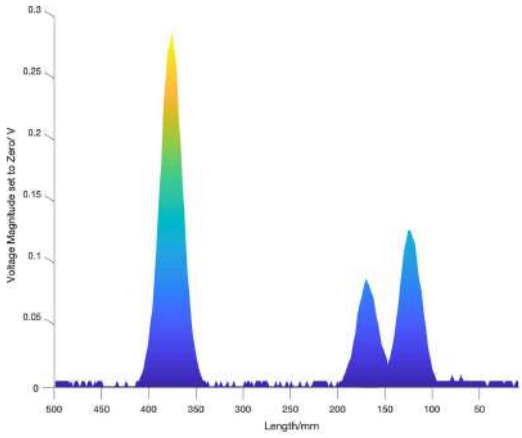

Figure 14: Magnitude of Data set to zero

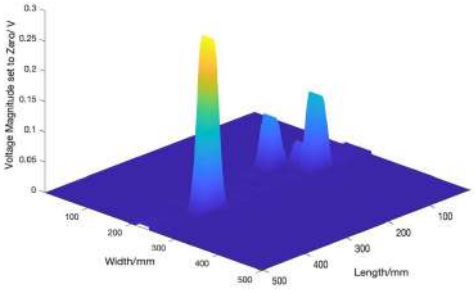

Figure 15: Magnitude of Data

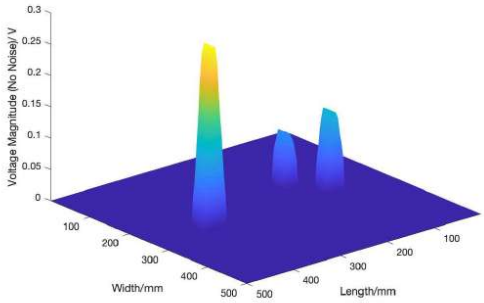

Figure 16: Data without noise

Figure 16 shows the same peaks highlighted in Figure 15 without the noise which produced miniature peaks in Figure 15.

```
function(TheY1)
Line No.
 55      Line1(Line1 < 0.02) = 0;    % Removes all low level values
 56      Line2(Line2 < 0.02) = 0;
 57      Line3(Line3 < 0.02) = 0;
```

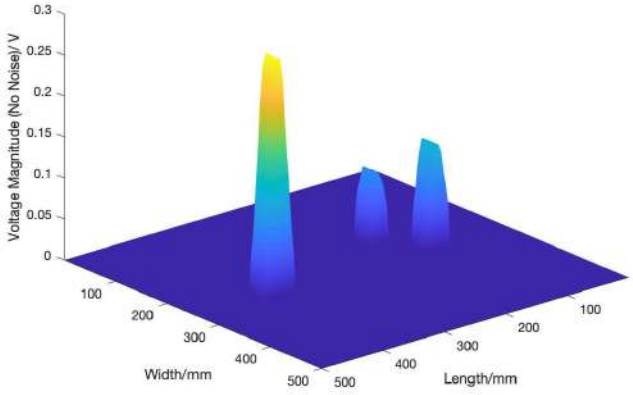

Figure 17: Magnet Peak Data

At this stage, all peaks have been clearly isolated. The PWM coordinate of the highest value of the highest peak is the stored along with which Hall effect scan row it is in. This value and a range of 320 PWMs either side are then set to zero. This equates to $\pm$10mm.

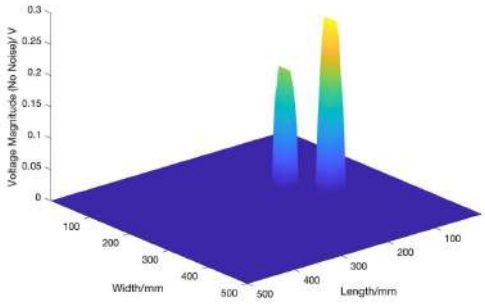

Figure 18: Magnet Peak Data -1 peak

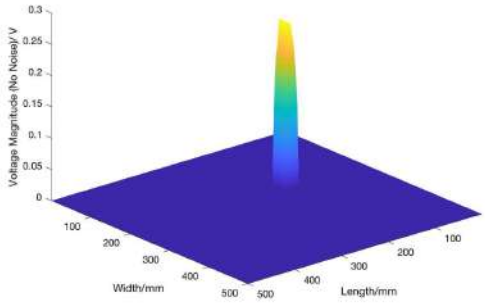

Figure 19: Magnet Peak Data -2 peaks

This process is continued in a while loop. Storing the coordinates of the largest peak and then removing it from the array. Once all peaks have been cleared, all values in the array will be equal to zeros. This marks the end of the data processing.

6 Software Development

After the development of the data processing, testing showed a problem. Magnets that were between two scanning passes would be picked up by both. The system would then initiate a cup drop on this magnet after both passes, double marking the cup.

```
function(TheY1)
Line No.
        %  Double drop avoidance break

       if pwmPeak2(1,RowOrdr2(1,V)) >= (DND2(1,2) -150)
          || pwmPeak2(1,RowOrdr2(1,V)) <= (DND2(1,2) +150)

           break
       end
```

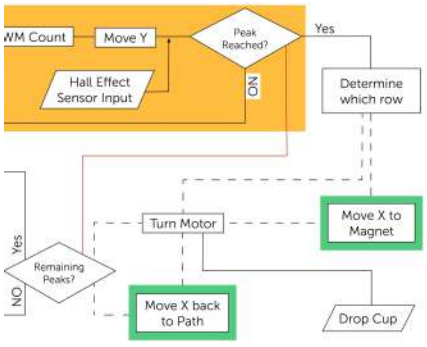

Figure 20: Software Flow Chart

This IF statement was imbedded in the functions for the second, third & forth scanning paths. This IF statement acted as a last minute cancelation. It would break out of the cup drop process if a magnets were sensed in row one and row three of the previous scan, at a similar PWM co-ordinate. The red line in Figure 18 represents the break made by the if statement. It by passes the process of moving to the magnet row and dropping a cup. Instead, the PWM count would continue to the next sensed peak and mark the next magnet. This addition directly addressed specification point 21, Avoid previously marked locations of magnets.

7 Reflection on Personal Learning Outcomes

The initial Learning outcomes were set at the beginning of the project and focused on developing my understanding of software development. This was selected as the learning outcome based on my limited prior experience. Software development was also considered to be a broader and more applicable skill than electronics, that could be applied in a range of fields in the future.

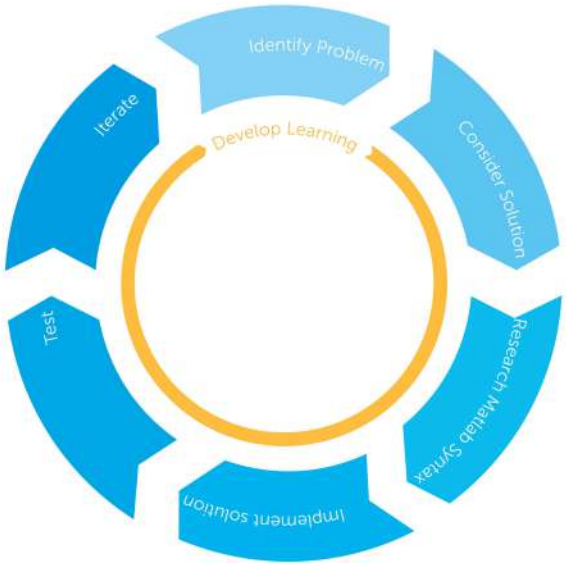

Figure 21: Cyclical learning Process

Figure 21 shows the cyclical learning process that was used throughout this project. Software development, from my understanding, is a continuous problem-solving exercise. Therefore, the first step was always problem identification. What am I trying to achieve? From there I would think of a logical solution based on mathematical understanding. In the majority of cases, I would then need to research how to translate my solution into Matlab and from there I could work on implementation. I would then test the solution and move on to the next problem or address any issues raised through testing. Using this method, I would consider my learning outcome a success as I definitely evolved my understanding of software development through this process. I would also consider the learning methodology to be sound as by the end we had a working software. Despite this successful outcome, there is much progression that could still be made. The model in Figure 21 is functional but fairly basic. It does not address any of the finer points of software development and this is something I certainly failed to grasp during the

process. In the future I will aim to apply more best practice methods for software development. Examples include:

· Repositories: Saving functioning code before delving in and changing everything. This one definitely caused problems at times.

· Development Methodologies: A more structured plan would help clarify the required software outcomes.

· Streamlining: Writing code in the most efficient way possible. I undeniably wrote more code than was absolutely necessary.

8 Conclusion

Once again to state Occam's Razor and the theory behind this project:
"The Simplest solution is almost always the best"

The solutions found for this task, both mechanical and technological, met aptly with this statement. The mechanical drivetrain allowed for simplification in both the software and hardware. The software itself was basic but performed as required based on the physical limitations of the problem.

The solution found was simple, in places perhaps too simple. The entire system relied heavily on the markers dropping 400mm and landing correctly, which you could called optimistic or perhaps even wishful thinking. The magnet location was accurate in one dimension only, which required a reliable PWM count. given the demonstration performance, this was perhaps not the most robust method.

I personally met my Learning outcome completely, though this is mainly due to me setting myself a very basic learning outcome. In future I will aim higher and would like to grasp a deeper understanding of efficient and robust software development.

9 References

Bradley, D., Bradley, D. and Loader, A. (2018). Mechatronics and the design of intelligent machine and systems. Boca Raton: Routledge.

Powell-Morse, A. (2019). V-Model: What Is It And How Do You Use It?. [online] Airbrake Blog. Available at: https://airbrake.io/blog/sdlc/v-model [Accessed 6 Dec. 2019].

10 Appendix - Matlab Code

```
%The Big one

% Configure
 clear all;

 a = arduino('/dev/tty.usbmodem14101','Mega2560','Libraries',
     {'RotaryEncoder','Servo', 'BathUniversity/StepperMotorAddOn'});

% Configuring Pins Y axis

 configurePin(a,'D22','DigitalOutput')   % 6:Enable Pin
 configurePin(a,'D23','DigitalOutput')   % 7:Direction Pin
 configurePin(a,'D6','PWM')              % 8:Pulse Pin
 configurePin(a,'D25','DigitalInput')    % 9:Terminal switch

% Write Pins

 writeDigitalPin(a,'D22', 0)        % Set Y Enable Pin to active low
 writeDigitalPin(a,'D23',0)         % Direction Y Pin 0:CW 1:CCW

% Configuring Pins X axis

 configurePin(a,'D32','DigitalOutput')   % 1:Enable Pin
 configurePin(a,'D33','DigitalOutput')   % 2:Direction Pin
 configurePin(a,'D7','PWM')              % 3:Pulse Pin
 configurePin(a,'D34','DigitalInput')    % 4:Terminal switch

% Write Y Pins

 writeDigitalPin(a,'D32',0)         % Set X Enable Pin to active low
 writeDigitalPin(a,'D33',0)         % Direction X Pin 0:CW 1:CCW

% Counter

 Ypwm = 0.5;
 Xpwm = 0.5;

% Configuring Hall Effect Pins

 configurePin(a,'A8','AnalogInput')
 configurePin(a,'A9','AnalogInput')
 configurePin(a,'A10','AnalogInput')

 TheOriginFinder(a,Ypwm, Xpwm);
```

```
[pwmPeak] = TheY1(a, Ypwm);

TheCupCollection(a, Xpwm);

[pwmPeak] = TheY42(a, Ypwm, Xpwm, pwmPeak);

TheX1(a, Xpwm);

[pwmPeak] = TheY43(a, Ypwm, Xpwm, pwmPeak);

TheX1(a, Xpwm);

TheY44(a, Ypwm, Xpwm, pwmPeak);
```

```
%% Scan Y1
function [ pwmPeak] = TheY1(a, Ypwm)

% Scan Variables
  hesArray  = zeros(3, 100);    % Hall effect array
 pwmArray  = zeros(1, 210);

 YDistance = 425;

 YPulse = (YDistance * 32);                 % Set number of pulses per mm

 YCounter  = rotaryEncoder(a,'D18','D19');
 writeDigitalPin(a,'D23',1)                 % Direction Y Pin 0:CW 1:CCW
 writePWMDutyCycle(a,'D6',Ypwm)             % Y speed controlle

 n = 1;
 p = 1;
 YCount = 0;

while YCount < YPulse

      yCount = readCount(YCounter);
      YCount = abs(yCount);

      hesArray(n,p)   = readVoltage(a,'A8');     % Hall Effect data
      hesArray(n+1,p) = readVoltage(a,'A9');
      hesArray(n+2,p) = readVoltage(a,'A10');

      pwmArray(n,p)   = YCount;

      p = p +1;
end

 writePWMDutyCycle(a,'D6',0)        % Y speed controlle

 Line1 = hesArray(n,:);             % Isolate lines from array
 Line2 = hesArray(n + 1,:);
 Line3 = hesArray(n + 2,:);

 avg1 = mode(Line1);                % sets to zero approximatly
 avg2 = mode(Line2);
 avg3 = mode(Line3);

 Line1 = Line1 - avg1;              % sets to zero by removing average
 Line2 = Line2 - avg2;
 Line3 = Line3 - avg3;

 Line1 = abs(Line1);                % Finds Magnitude of data
 Line2 = abs(Line2);
```

```
Line3 = abs(Line3);

Line1(Line1 < 0.02) = 0;           % Removes all values below clouds
Line2(Line2 < 0.02) = 0;
Line3(Line3 < 0.02) = 0;

Line1(Line1 > 2) = 0;              % Removes all values below clouds
Line2(Line2 > 2) = 0;
Line3(Line3 > 2) = 0;

Peak         = zeros(2,30);        % Sets inital zero arrays
PeakPosition = zeros(1,10);
pwmPeak      = zeros(2,15);
Row          = zeros(1,15);

Line(1,:) = Line1;
Line(2,:) = Line2;
Line(3,:) = Line3;

m = 1;                                  % m = Magnet no
P = 1;

 while max(Line1) > 0 || max(Line2) > 0 || max(Line3) > 0

      % Finds peaks
     [Peak(2,m),Peak(1,m)]     = max(Line(1,:));
     [Peak(2,m+1),Peak(1,m+1)] = max(Line(2,:));
     [Peak(2,m+2),Peak(1,m+2)] = max(Line(3,:));

     [PeakPosition(2,P), Row(1,P)] = max(Peak(2,(m:m+2)));
     PeakPosition(1,P) = Peak(1,(Row(1,P)+ (3*P - 3)));

     if max(Peak(2,(m:m+2))) == 0
         break
     end

     % Finds PWM value for Peak
     pwmPeak(1,P) = pwmArray(1,PeakPosition(1,P));

     Line(:,(abs(PeakPosition(1,P) - 9)):(PeakPosition(1,P) + 9)) = 0;

     m = m + 3;
     P = P + 1;

 end

 pwmPeak(2,:) = Row(1,:);

end
```

```
%% Scan X1

function [] = TheX1(a, Xpwm)

 XDistance = 90;

 XPulse = (XDistance * 32);        % Set number of pulses per mm
% Tollerance of +- 1mm

 XCounter = rotaryEncoder(a,'D20','D21');
 writeDigitalPin(a,'D33',1)        % Direction X Pin 0:CW 1:CCW
 writePWMDutyCycle(a,'D7',Xpwm)    % X speed controlle

 XCount = 0;

while  XCount <= XPulse

       xCount = readCount(XCounter);
       XCount = abs(xCount);
end

 writePWMDutyCycle(a,'D7',0)      % X speed controlle
```

```
%% Scan Y2   Same Functions for Y3 & Y4
function [pwmPeak] = TheY2(a, Ypwm, Xpwm)

% Scan Variables

 V = 1;
 n = 1;
 p = 1;
 B = 2;
 D = 1;

 TheXCounter2 = zeros(1,9);

 hesArray  = zeros(3, 100);    % Hall effect array
 pwmArray  = zeros(1, 210);
 DND3 = zeros(1,3);

  YCounter  = rotaryEncoder(a,'D18','D19');
  XCounter = rotaryEncoder(a,'D20','D21');

 [pwmpeak2, RowOrdr2] = sort(pwmPeak2(1,:),'descend');
  XCount = 0;


 %while n == 1
 while pwmpeak2(1,V) == 0 || n == 1

 DntDrop2 = 0;

 writeDigitalPin(a,'D23',1)          % Direction Y Pin 0:CW 1:CCW
 writePWMDutyCycle(a,'D6',Ypwm)      % Y speed control

 YCount = 0;

while abs(YCount) < (13600 - pwmpeak2(1,V))

      yCount = readCount(YCounter);
      YCount = abs(yCount);

      hesArray(n,p)   = readVoltage(a,'A8'); % Hall Effect data
      hesArray(n+1,p) = readVoltage(a,'A9');
      hesArray(n+2,p) = readVoltage(a,'A10');

      pwmArray(n,p)   = YCount;

      p = p + 1;

end
```

```
 writePWMDutyCycle(a,'D6',0)          % X speed control

 XDistance = 30;

 XPulse = (XDistance * 32);           % Set number of pulses per mm
 % Tollerance of +- 1mm

 while  abs(XCount) <= (XPulse + TheXCounter2(1,B))

   if pwmPeak2(2,RowOrdr2(1,V)) == 1

       %  Double drop avoidence breaks

      if pwmPeak2(1,RowOrdr2(1,V)) >= (DND2(1,1) -150)
         || pwmPeak2(1,RowOrdr2(1,V)) <= (DND2(1,1) +150)

%           DntDrop2 = 1;
          break

      end

      if pwmPeak2(1,RowOrdr2(1,V)) >= (DND2(1,2) -150)
         || pwmPeak2(1,RowOrdr2(1,V)) <= (DND2(1,2) +150)

%           DntDrop2 = 1;
          break

      end

      if pwmPeak2(1,RowOrdr2(1,V)) >= (DND2(1,3) -150)
         || pwmPeak2(1,RowOrdr2(1,V)) <= (DND2(1,3) +150)

%           DntDrop2 = 1;
          break

      end

      %----%

     writeDigitalPin(a,'D33',0)       % Direction X Pin 0:CW 1:CCW
     writePWMDutyCycle(a,'D7',Xpwm)    % X speed control

   end

   if pwmPeak2(2,RowOrdr2(1,V)) == 3

      writeDigitalPin(a,'D33',1)        % Direction X Pin 0:CW 1:CCW
      writePWMDutyCycle(a,'D7',Xpwm)    % X speed control
```

```
% Sets up array to prevent double drop
      DND3(1,D) = pwmPeak2(1,RowOrdr2(1,V));

      D = D +1;

   end

   if pwmPeak2(2,RowOrdr2(1,V)) == 2
       break
   end

   if pwmPeak2(2,RowOrdr2(1,V)) == 0
       break
   end

   xCount = readCount(XCounter);

   XCount = abs(xCount);

 end

 writePWMDutyCycle(a,'D7',0)

    B = B+1;
   TheXCounter2(1,B:B+3) = XCount;

if DntDrop2 == 0


      %% Cup Drop
    TheStepperMotor(a)

    XDistance = 30;

    XPulse = (XDistance * 32);   % Set number of pulses per mm
    % Tollerance of +- 1mm

   while  abs(XCount) <= (XPulse + TheXCounter2(1,B))

      if pwmPeak2(2,RowOrdr2(1,V)) == 2
          break
      end

      if pwmPeak2(2,RowOrdr2(1,V)) == 1

         writeDigitalPin(a,'D33',1)    % Direction X Pin 0:CW 1:CCW
         writePWMDutyCycle(a,'D7',Xpwm)   % X speed control
      end
```

```
      if pwmPeak2(2,RowOrdr2(1,V)) == 3

         writeDigitalPin(a,'D33',0)              % Direction X Pin 0:CW 1:CCW
         writePWMDutyCycle(a,'D7',Xpwm)          % X speed control
      end

      if pwmPeak2(2,RowOrdr2(1,V)) == 2
         break
      end

      if pwmPeak2(2,RowOrdr2(1,V)) == 0
          break
      end

     xCount = readCount(XCounter);

     XCount = abs(xCount);

   end

         writePWMDutyCycle(a,'D7', 0)     % X speed control

              V = V + 1;

end

  B = B + 1;
  TheXCounter2(1,B:B+3) = XCount;

 end

 writePWMDutyCycle(a,'D6',Ypwm)      % Y speed control

while abs(YCount) <= 13600

      yCount = readCount(YCounter);
      YCount = abs(yCount);

      hesArray(n,p)   = readVoltage(a,'A8');  % Hall Effect data
      hesArray(n+1,p) = readVoltage(a,'A9');
      hesArray(n+2,p) = readVoltage(a,'A10');

      pwmArray(n,p)   = YCount;

      p = p + 1;

end

 writePWMDutyCycle(a,'D6',0)      % Y speed control
```

```
Line7 = hesArray(n,:);          % Isolate lines from array
Line8 = hesArray(n + 1,:);
Line9 = hesArray(n + 2,:);

avg7 = mode(Line7);             % sets to zero approximatly
avg8 = mode(Line8);
avg9 = mode(Line9);

Line7 = Line7 - avg7;          % sets to zero by removing average
Line8 = Line8 - avg8;
Line9 = Line9 - avg9;

Line7 = abs(Line7);         % sets to zero by removing average
Line8 = abs(Line8);
Line9 = abs(Line9);

Line7(Line7 < 0.04) = 0;       % Removes all values below clouds
Line8(Line8 < 0.04) = 0;
Line9(Line9 < 0.04) = 0;

Line7(Line7 > 2) = 0;       % Removes all values below clouds
Line8(Line8 > 2) = 0;
Line9(Line9 > 2) = 0;

Peak3         = zeros(2,30);        % Sets inital zero arrays
PeakPosition3 = zeros(1,10);
pwmPeak3      = zeros(1,10);
Row3          = zeros(1,100);
                                                % m = Magnet no.

Line3(1,:) = Line7;
Line3(2,:) = Line8;
Line3(3,:) = Line9;

m = 1;
P = 1;

 while max(Line7) > 0 || max(Line8) > 0 || max(Line9) > 0


       % Finds peaks
     [Peak(2,m),Peak3(1,m)] = max(Line3(1,:));
     [Peak(2,m+1),Peak3(1,m+1)] = max(Line3(2,:));
     [Peak(2,m+2),Peak3(1,m+2)] = max(Line3(3,:));

     [PeakPosition(2,P), Row3(1,P)] = max(Peak3(2,(m:m+2)));
     PeakPosition(1,P) = Peak3(1,(Row3(1,P)+ (3*P - 3)));
```

```
      % peak row 1 is the 394 coordinate
      % Row is the row that it should be droppped in
      if max(Peak3(2,(m:m+2))) == 0 || P == 100

          break
      end

      % Finds PWM value for Peak
      pwmPeak3(1,P) = pwmArray(1,PeakPosition3(1,P));

     %pwm peak is the exact coordinate

       %Clears out old peaks
      Line3(:,(abs(PeakPosition3(1,P) - 15)):(PeakPosition3(1,P) + 15)) = 0;

      m = m + 3;
      P = P + 1;

  end

   [~, c] = size(pwmPeak3);
    pwmPeak3(2,:) = Row3(1,c);


end
```