|  |
| --- |
| University of Guelph: School of Engineering |
| ENGG\*3490: Mechatronics Systems Design Final Project |
| Prof: M. Biglarbegian Ph.D., P.Eng. |

|  |
| --- |
| Rookie Ayson, Mohammed Safwan, Ammar Funginis, Wadih Khatib, Jake O’Neill, Saif Matar  [Date] |

Abstract

The main goal of this project was to design a nut sorting mechanism capable of separating four different types of nuts. Given distinct specifications like build materials and dimensions the mechanism is required to perform its function without violating anything that has been specified. The system also required to be smart enough to differentiate more than just nuts, in this project coins were used to throw off the sensors and test the robustness of the design. With the use of different sensors tested in labs the machine was provided enough input information to sense and distribute the items that were being fed. In this report, a list of all the different sensors and their functions will be included. It outlines their primary functions and why they’re required to add functionality to the machine. A simple flowchart also explains the conditional statements the machine runs through in order to determine what has been fed. Experimental results provide proof of testing and shows the efficiency of the design. Further improvements and recommendations can be implemented using these results as the basis. Therefore, machine design will always be subjected to change and the scope of the project will further broaden as each modification is incorporated.

Table of Contents

[Introduction 2](#_Toc4687261)

[Methodology 2](#_Toc4687262)

[Experiments and Results 2](#_Toc4687263)

[Conclusion and Recommendations 2](#_Toc4687264)

[Future Scope 2](#_Toc4687265)

[References 2](#_Toc4687266)

[Appendix 2](#_Toc4687267)

# Introduction

The purpose of this project was to design a mechatronic nut-sorting system that can categorize and sorting different types of nuts. There are four distinct types of nuts and three of each type that must be separated into their respective containers. The nut types to be sorted are small steel, medium brass, large brass, and large nylon nuts, and the order of nuts input into the machine will be randomized. Additionally, 1 toonie, 2 loonies, and 4 quarters will be placed into the system at random intervals. The system should be robust enough to distinguish the coins from the nuts so that no coins are sorted into the nut containers. The design should be mechatronics based, making use of sensors as well as actuators to sort the nuts, rather than mechanically or manually based.

# Methodology

Sensors used include the FSR 406 single-zone Force Sensing Resistor, the Sharp GP2Y0A41SK0F Analog Distance Sensor, and the SparkFun 9DOF sensor stick which includes an accelerometer, a magnetometer, and a gyroscope. The force sensor works by showing a decrease in resistance with an increase in applied force and is used to differentiate the large brass nut from everything else. The distance sensor works by emitting an IR beam that reflects off an object and returns to the receiver, outputting a voltage. The greater the distance between the sensor and the object, the lower the output voltage. This sensor is used to differentiate the large nuts from the small nuts as well as the coins. The 9DOF sensor only makes use of the magnetometer which works using three magneto-resistive materials arranged in perpendicular axes (x,y,z). Changes in the local magnetic field are detected by measuring the changes in current flowing through the three magneto-resistive materials. This sensor is used to differentiate the steel nuts and coins from the other nuts. Additionally, a slot was made that allows coins to pass through, leaving them unsorted. A flowchart of the system can be found in Figure 1 below.



Figure 1: Sorting Mechanism Flowchart

# Experiments and Results

## Force Sensor

In order to calibrate the force sensor, the first task was to determine the method of operation for it. Consulting the datasheet for the force sensor, it was observed that unloaded the force sensor acts as a very large resistor. As such using the breadboard and Arduino an analog signal was output through the force sensor read from one of the Arduino’s analog pins. During testing it was shown that only the large brass nut was heavy enough to produce enough pressure to trigger the sensor as seen in Figure 2 below.

Figure 2: Force sensor readings for all nuts

To address this the group decided to use the force sensor to determine whether the large brass nut was placed into the machine.

## Magnetic Sensor

The magnetic sensor was used to determine whether a magnetic material, such as steel or nickel, was placed into the machine. This would be used to detect whether a steel nut or coin was placed into the machine and would activate the machines arm to point in the correct direction. This direction was selected to filter out coins through an opening in the machine’s chute, whilst allowing steel nuts to continue on to the cup. During calibration of the sensor it was discovered that only one axis of the magnetic sensor was changing significantly enough to be used an indicator. As such the program would only consider large changes to the sensors x-axis reading when determining which object was placed.

## Range Finder

Lastly, the range finder was used to determine if the large nylon nut was placed into the machine. As the nylon nut was taller than all other objects, excluding the large brass nut, the range finder was attached in a manner that would allow the machine to read the height of the objects passing through. Figure 3 below illustrates the read outs from the range finder when placing each of the different objects into the machine.

Figure Range Finder Measurements for different objects

## Results

During demonstration the machine was able to correctly sort 11 of the 12 objects in under 70 seconds. Whilst the timing was admirable for a first build, the sorting results were disappointing. The source of error was identified as the inaccurate nature of the range finder. The range fined works by triangulating an emitted infrared beam. By emitting an infrared beam at an angle, the sensor determines the sharpness of the angle of the reflected beam and outputs the reading as an analog signal. Due to the placement and size of the infrared range finder, there were many objects the beam can be reflected off, some within the range of values that would trigger the sorting mechanism for the large nylon nut. The team have determined that whilst the range finder is useful, its tendency to misread information is an issue. As such the team has decided to replace the range finder with a color sensor. It is predicated that the color sensor is a more viable option as the nylon nut is a very different color to the surrounding wood. In addition, the color sensor is much smaller and has a closer operating range that will allow the team to place the sensor very close to the conveyer system. This will reduce the field of view of the sensor, measuring a smaller area which will mitigate error.

Whilst the speed of the machine is admirable it could be improved. The main limiting factor is the use of a polling method on the sensor readings. The sensors readings are read once during every loop of the system. The values of the readings are then used in a truth table to determine the correct course of action. This means that the conveyer system must be slow enough to allow the sensors to obtain an accurate snapshot of their environment. As such the pulse width modulation frequency of the motor was set low enough to allow the object to stay on the sensing platform for at least three cycles. To address this, modifications must be made to the program to change how action is determined. Rather than rely on truth statements to make decisions, the machine will be reprogrammed to operate as a real-time system. This will rely on system interrupts when a sensor reads a key value. In general, the interrupt method is faster than the polling method as the microcontroller is not monitoring signals, but instead awaits a signal from the sensors.

# Conclusion and Recommendations

In conclusion the design of the nut sorter was unique and functional 92% of them time according to the trials performed. The Nut sorter was able to sort 11 out of the 12 nuts that were tested along with the 4 coins that the device did not register. All these nuts and coins were sorted under 70 seconds. There 4 different types of nuts with different materials, size and weight. A combination of pressure sensor, range finder and magnetic sensors were used to identify the nuts or coins. The speed of the conveyer belt had to limited since the pressure sensor and magnetic sensors needed some extra time to assess the properties of the nuts.

While the design of the nut sorter was successful, achieving 11/12 sorting, and average time to complete the sorting, it can be improved to get an overall better result. A conveyer belt was used, which takes a linear path to move the nuts. This is problematic if speed is a constraint for the design. Having a conveyer belt limits the user to placing 1 nut at a time, which drastically slows down the time it takes to complete the task. One possible way to mitigate this issue to have multiple lines running at the same to allow for multiple nuts to get sorted at the same time. However, it is not believed that this is a practical solution as it would take more materials to achieve and more sensors and more control. Overall, it would be more complicated and would cost more to implement. With a conveyer belt, the belt material must be considered. It needs to have enough friction so the nuts would not slide, but not too much. The belt also needs a perfect amount of slack so that it would still be able to roll on the rollers but allow for the big brass nut to activate the pressure sensor that is located underneath the belt. Overall, the design has brought upon many challenges that could be mitigated with a different design. Many other groups utilized a turntable design that allows multiple nuts to be fed at the same time or consecutively, which overall would lead to a faster design. Should the machine be redesigned from scratch, a turntable design or a similar design to achieve faster results similar to other groups. The sensors that would be used could be to measure the height difference, conductivity instead of magnetism since the magnetism sensor is very sensitive and needs to be calibrated frequently.

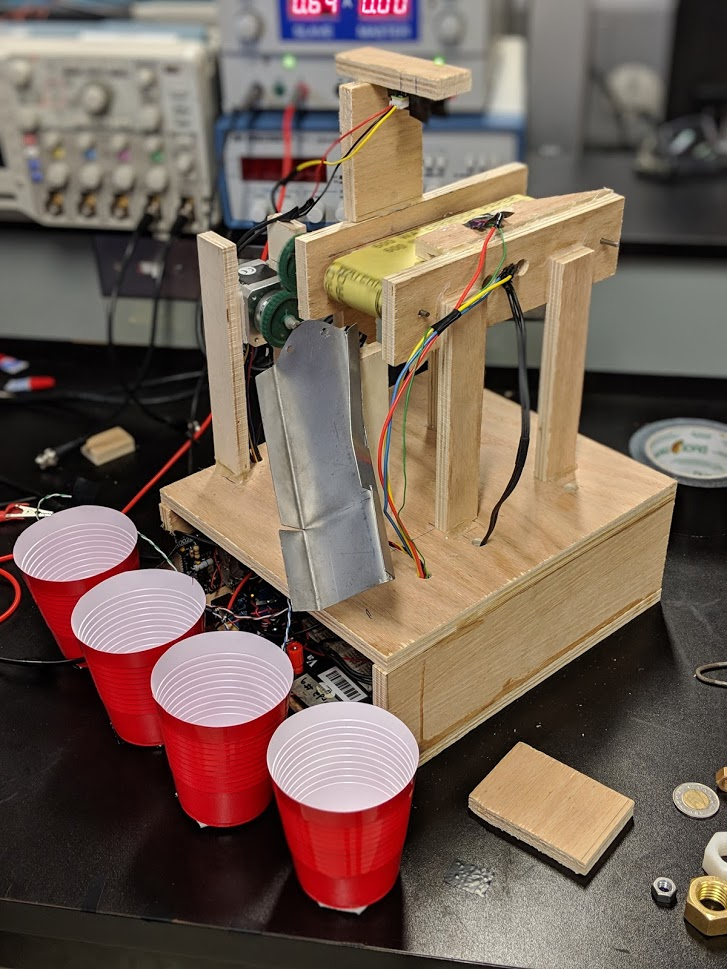
In terms of the design itself, wood was used to build the entire structure of the design. The team would not change the material used it is an economical option with enough strength to function without failure. However, the joining method would change. In future, more time would be allocated towards the design of the mechanism to ensure screws, bolts and nuts, and fasteners were used instead of glue to hold the mechanism together. Overall, this results in a cleaner, more professional look that also allows for change when needed. Once the parts are glued it becomes hard to take off to make any changes.

# Future Scope

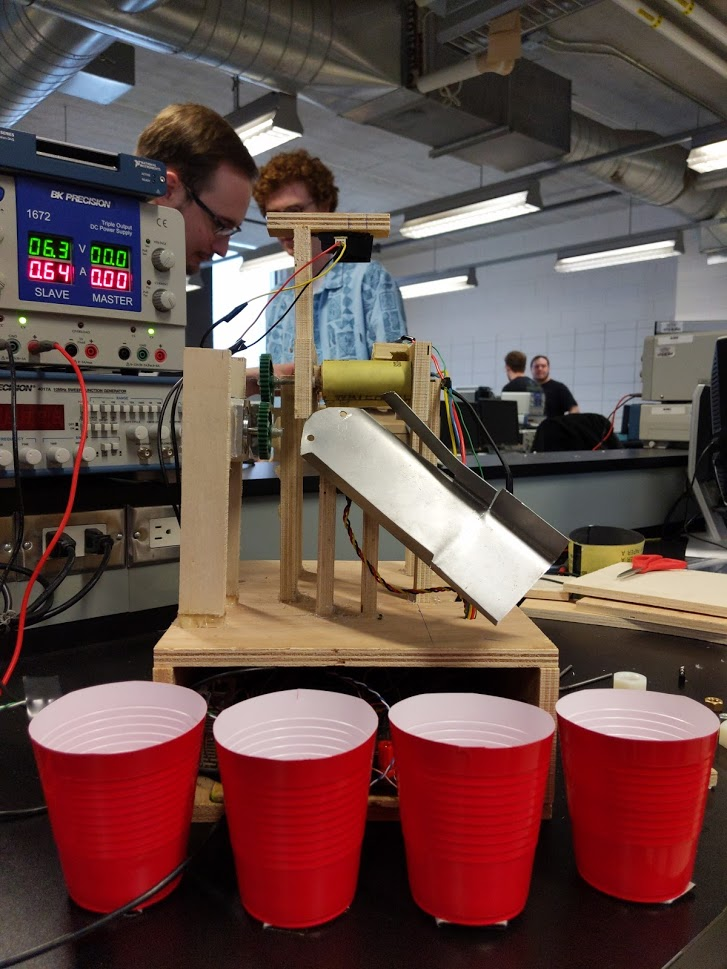
For future improvement, the group discussed to make use of different sensors that are more stable and efficient. The change of sensors is essential, as the system can be faster with the use of a sensor that can detect color. Another addition to the system can be using a better covering belt that does not lag (loose friction to the rollers) The main performance increase that the group wants to target is the speed, the system finished the sorting of 12 different objects in 59 seconds, it will be ideal to decrease it to range of 20 to 30 seconds. Lastly, the system will be updated to sort anything that is not detected as a nut to a dumpster cup.

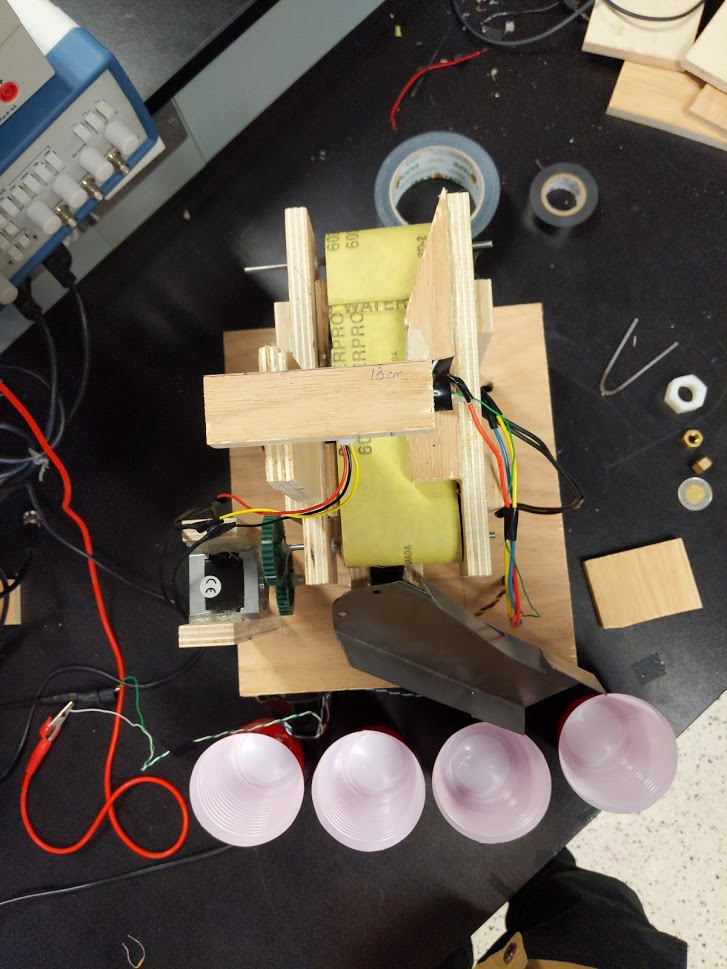
# References

# Appendix









Code:

#include <PWM.h>

#include <Wire.h>

#include <Servo.h>

#define MAGN\_ADDRESS ((int16\_t) 0x1E) // 0x1E = 0x3C / 2

#if ARDUINO >= 100

#define WIRE\_SEND(b) Wire.write((byte) b)

#define WIRE\_RECEIVE() Wire.read()

#else

#define WIRE\_SEND(b) Wire.send(b)

#define WIRE\_RECEIVE() Wire.receive()

#endif

boolean output\_errors = false;

#define HW\_\_VERSION\_CODE 10736

#define MAGN\_X\_MIN ((float) -600)

#define MAGN\_X\_MAX ((float) 600)

#define MAGN\_Y\_MIN ((float) -600)

#define MAGN\_Y\_MAX ((float) 600)

#define MAGN\_Z\_MIN ((float) -600)

#define MAGN\_Z\_MAX ((float) 600)

#define MAGN\_X\_OFFSET ((MAGN\_X\_MIN + MAGN\_X\_MAX) / 2.0f)

#define MAGN\_Y\_OFFSET ((MAGN\_Y\_MIN + MAGN\_Y\_MAX) / 2.0f)

#define MAGN\_Z\_OFFSET ((MAGN\_Z\_MIN + MAGN\_Z\_MAX) / 2.0f)

#define MAGN\_X\_SCALE (100.0f / (MAGN\_X\_MAX - MAGN\_X\_OFFSET))

#define MAGN\_Y\_SCALE (100.0f / (MAGN\_Y\_MAX - MAGN\_Y\_OFFSET))

#define MAGN\_Z\_SCALE (100.0f / (MAGN\_Z\_MAX - MAGN\_Z\_OFFSET))

Servo servo\_test; //initialize a servo object for the connected servo

//int angle = 180;

float magnetom[3];

int servoPin = A8;

int flexiForcePin = A0;

int pin = 12;

int enPin = 26;

int dirPin = 32;

int IRSensor = A2;

int freq = 1250;

int num\_magn\_errors = 0;

//int IRpin = 5;

//int pos = 2;

//long previousMillis = 0;

//long interval = 3000;

void setup() {

pinMode(pin, OUTPUT);

pinMode(enPin, OUTPUT);

pinMode(dirPin, OUTPUT);

digitalWrite(enPin, HIGH);

digitalWrite(dirPin, HIGH);

pinMode(IRSensor, INPUT);

timerStart();

SetPinFrequencySafe(pin, freq);

pwmWrite(pin, 254);

servo\_test.attach(servoPin);

servo\_test.write(0);

Magn\_Init();

Serial.begin(9600);

}

void loop() {

unsigned long currentMillis = millis();

int flexiForceReading = analogRead(flexiForcePin);

Read\_Magn();

Serial.println(magnetom[0]);

if(flexiForceReading > 150){

moveServo(60);

// Serial.print("POS: ");

// Serial.println(servo\_test.read());

// Serial.println("Big Brass Nut");

delay(2750);

moveServo(0);

}

else if(magnetom[0] >= 750 || magnetom[0] <= 620){

moveServo(90);

// Serial.print("POS: ");

// Serial.println(servo\_test.read());

// Serial.println("Steel Nut");

delay(2750);

moveServo(0);

}

else{

int volts = analogRead(IRSensor);

Serial.print("Range: ");

Serial.println(volts);

if(volts >= 340 && volts <= 370){

moveServo(30);

// Serial.print("POS: ");

// Serial.println(servo\_test.read());

// Serial.println("Nylon Nut");

delay(2250);

moveServo(0);

volts = 0;

}

else{

moveServo(0);

// Serial.print("POS: ");

// Serial.println(servo\_test.read());

// Serial.println("Anything Else");

delay(1);

}

}

delay(250); //just here to slow down the output for easier reading

}

void moveServo(int location){

servo\_test.write(location);

}

void timerStart(){

InitTimersSafe();

}

void Magn\_Init()

{

Wire.beginTransmission(MAGN\_ADDRESS);

WIRE\_SEND(0x02);

WIRE\_SEND(0x00); // Set continuous mode (default 10Hz)

Wire.endTransmission();

delay(5);

Wire.beginTransmission(MAGN\_ADDRESS);

WIRE\_SEND(0x00);

WIRE\_SEND(0b00011000); // Set 50Hz

Wire.endTransmission();

delay(5);

}

void Read\_Magn()

{

int i = 0;

uint8\_t buff[6];

Wire.beginTransmission(MAGN\_ADDRESS);

WIRE\_SEND(0x03); // Send address to read from

Wire.endTransmission();

Wire.beginTransmission(MAGN\_ADDRESS);

Wire.requestFrom(MAGN\_ADDRESS, 6); // Request 6 bytes

while(Wire.available()) // ((Wire.available())&&(i<6))

{

buff[i] = WIRE\_RECEIVE(); // Read one byte

i++;

}

Wire.endTransmission();

if (i == 6) // All bytes received?

{

// 9DOF Razor IMU SEN-10125 using HMC5843 magnetometer

#if HW\_\_VERSION\_CODE == 10125

// MSB byte first, then LSB; X, Y, Z

magnetom[0] = -1 \* (int16\_t)(((((uint16\_t) buff[2]) << 8) | buff[3])); // X axis (internal sensor -y axis)

magnetom[1] = -1 \* (int16\_t)(((((uint16\_t) buff[0]) << 8) | buff[1])); // Y axis (internal sensor -x axis)

magnetom[2] = -1 \* (int16\_t)(((((uint16\_t) buff[4]) << 8) | buff[5])); // Z axis (internal sensor -z axis)

// 9DOF Razor IMU SEN-10736 using HMC5883L magnetometer

#elif HW\_\_VERSION\_CODE == 10736

// MSB byte first, then LSB; Y and Z reversed: X, Z, Y

magnetom[0] = -1 \* (int16\_t)(((((uint16\_t) buff[4]) << 8) | buff[5])); // X axis (internal sensor -y axis)

magnetom[1] = -1 \* (int16\_t)(((((uint16\_t) buff[0]) << 8) | buff[1])); // Y axis (internal sensor -x axis)

magnetom[2] = -1 \* (int16\_t)(((((uint16\_t) buff[2]) << 8) | buff[3])); // Z axis (internal sensor -z axis)

// 9DOF Sensor Stick SEN-10183 and SEN-10321 using HMC5843 magnetometer

#elif (HW\_\_VERSION\_CODE == 10183) || (HW\_\_VERSION\_CODE == 10321)

// MSB byte first, then LSB; X, Y, Z

magnetom[0] = (((uint16\_t) buff[0]) << 8) | buff[1]; // X axis (internal sensor x axis)

magnetom[1] = -1 \* (int16\_t)(((((uint16\_t) buff[2]) << 8) | buff[3])); // Y axis (internal sensor -y axis)

magnetom[2] = -1 \* (int16\_t)(((((uint16\_t) buff[4]) << 8) | buff[5])); // Z axis (internal sensor -z axis)

// 9DOF Sensor Stick SEN-10724 using HMC5883L magnetometer

#elif HW\_\_VERSION\_CODE == 10724

// MSB byte first, then LSB; Y and Z reversed: X, Z, Y

magnetom[0] = (int16\_t)((((uint16\_t) buff[0]) << 8) | buff[1]); // X axis (internal sensor x axis)

magnetom[1] = -1 \* (int16\_t)(((((uint16\_t) buff[4]) << 8) | buff[5])); // Y axis (internal sensor -y axis)

magnetom[2] = -1 \* (int16\_t)(((((uint16\_t) buff[2]) << 8) | buff[3])); // Z axis (internal sensor -z axis)

#endif

}

else

{

num\_magn\_errors++;

if (output\_errors) Serial.println("!ERR: reading magnetometer");

}

}