DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SENIOR DESIGN II REPORT

SPRING 2023

ELECTRIC VEHICLE WIRELESS CHARGING ALIGNMENT SYSTEM

TEAM 2

Christopher Prasad	6187088	cpras004@fiu.edu
Andy Alvarez	6140523	aalva533@fiu.edu
Rodolfo Ramos	3317835	rramos022@fiu.edu
Ivan Mendoza	6181776	imend041@fiu.edu
Maximiliano Mauna	5919959	mmaun003@fiu.edu

Mentor: Dr. Arif Sarwat / Milad Behnamfar

April
2023

Table of Contents

I. PROJECT DESCRIPTION

Engineering often involves applying complex mathematical equations and heavy scientific theory to solve real-world problems. However, engineers must first define the problem and identify who is affected by it, along with the importance of solving it. This section will describe the problem the team must solve. Inductive Power Transfer (IPT) is a technology that enables the transfer of power from one system to another across a relatively large air gap between two loosely coupled coils with no physical contact. IPT provides a clean and safe way of transferring power.

However, the current obstacle associated with this technology is the need for more alignment between the coils. Our project is about designing an electric vehicle alignment mechanism that will detect a charging coil under the electric vehicle. It will then align the transmitter coil directly below the charging coil allowing maximum power transfer and improving the wireless charging efficiency. The alignment mechanism should also be quick and fit under a standard vehicle chassis. To increase the effectiveness of the wireless charging alignment system, this project aims to design and develop an alignment mechanism for electric vehicles that will detect and align the transmitter coil with the charging coil, maximizing power transfer efficiency. Our research intends to design an electric vehicle alignment mechanism to detect the charging coil underneath the electric vehicle and align the transmitter coil directly below the charging coil. The alignment mechanism should be quick and small enough to fit beneath a typical vehicle chassis. Our ultimate objective is to create a magnetic field detection and tracking alignment mechanism to enable effective Inductive Power Transfer (IPT) for electric vehicles.

II. DESIGN PROCESS

Chassis Design

The Chassis is made of a Galvanized Steel Frame that goes around in an octagonal shape. This frame is strong enough to support the charging coil and is a solid foundation for all the other parts that would be implemented.

Fig. 1. Galvanized Steel Frame

The frame then is supported additionally by another bar in the middle that will contribute to better stability and strength. This steel bar will also hold the motors. The motors are off brand 12V motors that have high torque. Because of the torque, it can handle a large sum of weight. With the 4-inch size of the wheels on the motors, the total platform can haul a little over 35 pounds. The downside of this type of motor is that it is not very fast, but it is still able to push the required amount of weight.

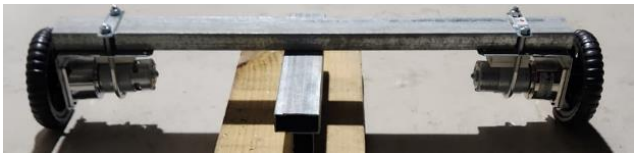


Fig. 2. Steel Bar that also holds the Motors

Fig. 3. Galvanized Steel Bar and Frame put together.

This frame will support a PVC platform. This platform will hold the charging coli and the sensor. Underneath this platform, another PVC platform was created to house future components such as

battery packs and microcontrollers. The frame was supported by two wheels powered by 12V motors and 4 Caster Wheels. The caster wheels give flexibility to move and allow for good stability while providing a firm foundation.

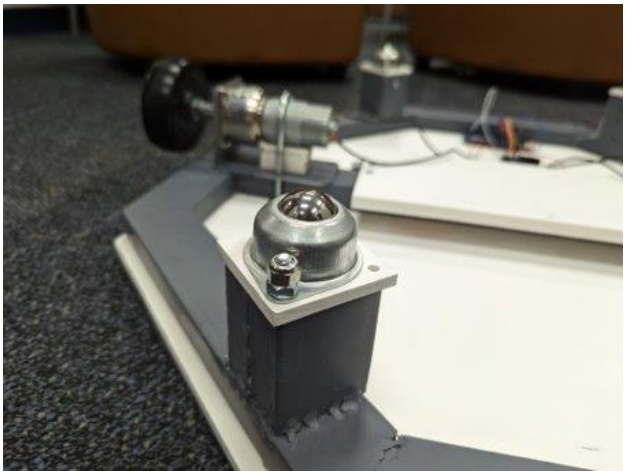


Fig. 4. Caster Wheels used to support the frame.

All these components came together to get the current version of the alignment system's chassis. This chassis is very strong and durable. It has the space and strength to hold the charging coil and all the future components. The overall chassis weighs around 30 pounds with the charging coil.

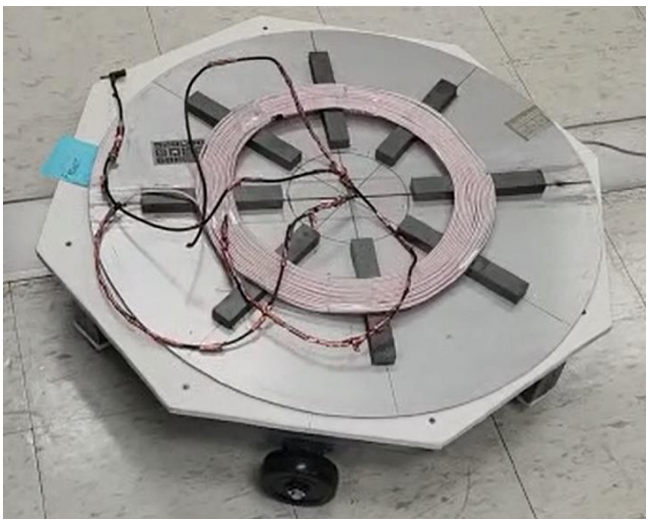


Fig. 5. Completed Chassis

Circuit Design

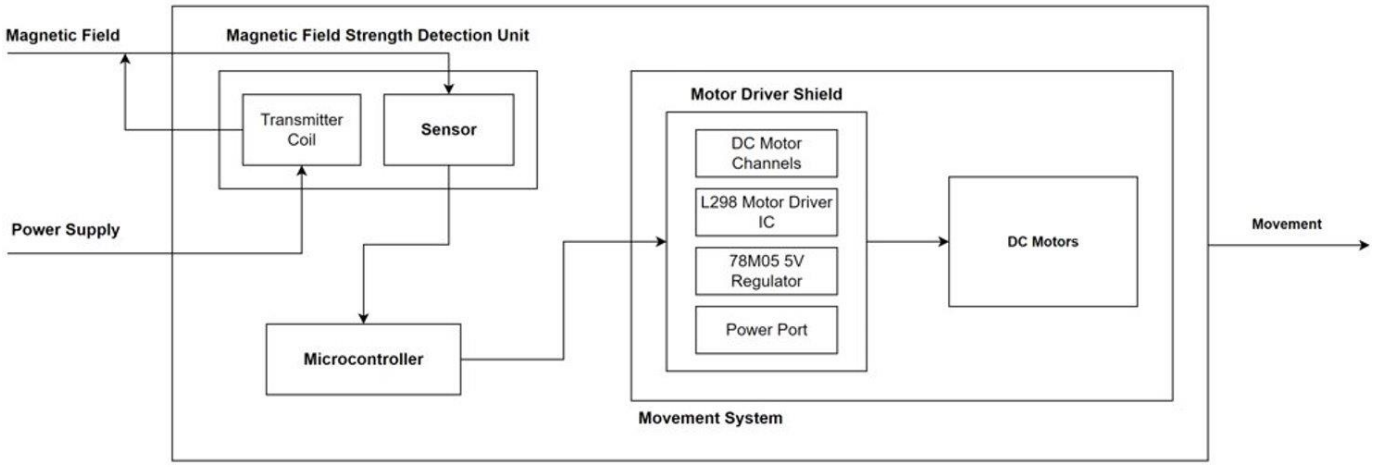


Fig. 6. Block Diagram

The entire system takes in two inputs and has one output. An existing magnetic field will be taken as an input to the magnetic field strength detection unit. The transmitter coil within the detection unit will use its own power supply to contribute to the strength of the existing magnetic field. The microcontroller receives magnetic field strength values from the hall effects sensor, which it then uses to perform calculations and logic on those values and outputs motor instructions. Since the motors cannot understand these instructions, the motor shield serves as the interface that translates these instructions for the motors to adjust the chassis position accordingly.

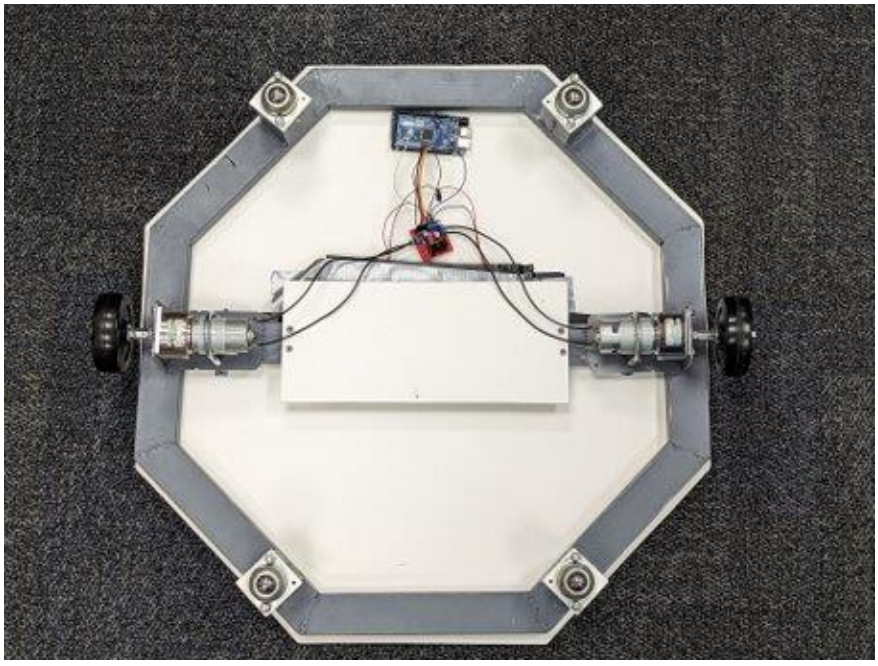


Fig. 7. Applied Block Diagram

Onto the chassis, two 12V motors were fitted onto the sides to propel the chassis into the desired directions. These motors were connected to the 3A L298N motor shield. This motor shield is strong enough to support the combined 24V battery pack is giving to the motors. The motor shield also allows for better controllability of the individual motors by providing induvial settings for direction and speed of each motor. This motor shield takes power from the power banks and powers the motor.

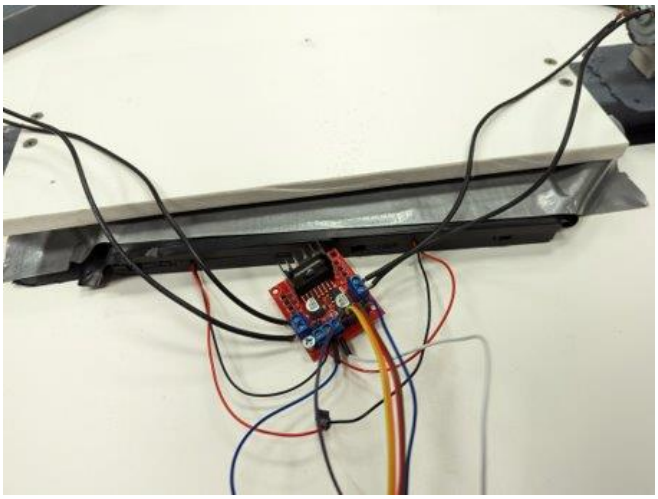


Fig. 8. Motor Shield and Hidden Power Bank

The motor shield gets instructions from the Arduino Mega 2560 Microcontroller to move the motors according to the implemented algorithm. The Mega is more than capable of executing the instructions that we give to it. Since it is a larger controller, it has the space to accommodate for additional features that could be implemented in the future such as more sensors.

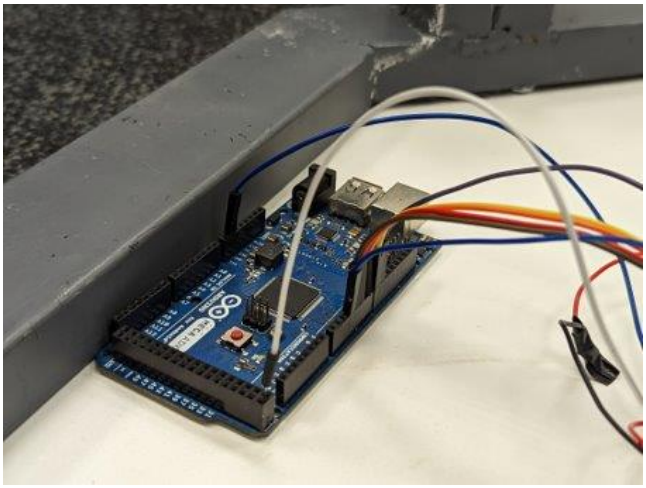


Fig. 9. Arduino Mega 2560 Microcontroller

The Arduino Mega 2560 Microcontroller processes logic based on the sensor values that is obtained through the hall effect sensor that is attached to the top of the alignment system. The hall

effect sensor is able to read the values from the magnetic field that the coils produce and accurately gives those values to the Arduino Mega 2560 to use.

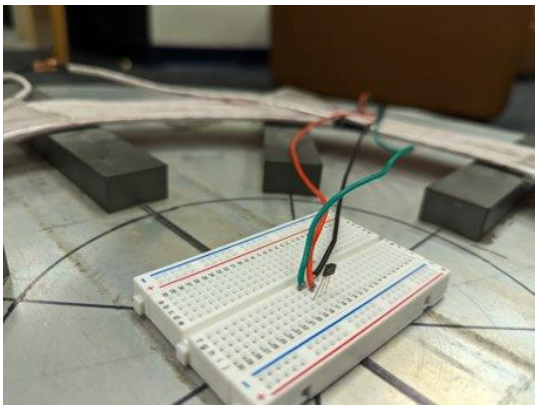


Fig. 10. Hall-Effect Sensor

III. ALGORITHM

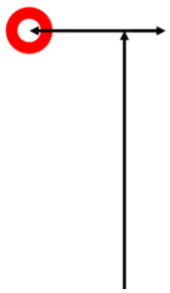
This section focuses on a highly specialized algorithm that is specifically designed to control the EV Wireless Charging Alignment System. This algorithm is critical in aligning the transmitter coil to the receiver coil to maximize their power transfer. This algorithm's unique and complex nature demands exceptional attention and understanding to ensure the effective functioning of the charging device. This section will delve deeply into the inner workings of this algorithm and explore its intricacies and nuances to gain a comprehensive understanding of its capabilities and limitations. It will explain the algorithm in detail, including its functionality, applications, and notable characteristics. By providing an in-depth analysis of this algorithm, this section aims to provide a comprehensive understanding of its workings, potential, and limitations and to highlight its importance in the field of electric vehicle wireless charging and its potential to revolutionize the industry.

This section also explains the algorithm that was fully developed and tested. The concept for this algorithm was designed to be as straightforward as possible to follow a test-driven development (TDD) and Agile approach, even though no test harnesses were used in conglomeration with the Arduino integrated development environment (IDE). This simplicity has several advantages with respect to the goal and expectations of this product as well as the current development stage. The following are some general goals and expectations for the project that drove the decision to take a simplistic approach and were taken into consideration when developing the algorithms.

Minimum Viable Product (MVP): An MVP is a concept from Agile scrum that refers to a product that has just enough features to satisfy the needs of early customers and, more importantly, gives them something to provide feedback on to shape the future of the product.

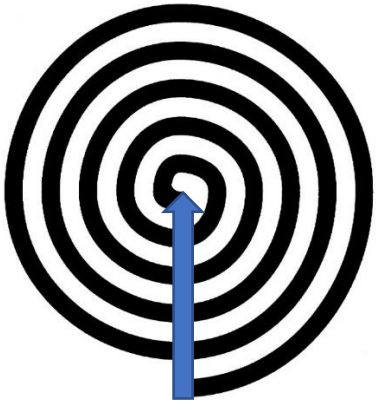
Developed Algorithms:

1) *Forward Single-Axis: The algorithm that is currently used and tested in the laboratory setting. This algorithm, when powered on, tells the motors to move forward. The alignment system will then move forward until a hardcoded setpoint value is received from the sensor. When the microcontroller gets the setpoint value from the sensor, it will tell the motors to stop. This only goes in one access.*
2) *Double Single-Axis Alignment: This algorithm was developed but was not able to be tested in the laboratory setting. This algorithm will tell the alignment system to move forward until the hardcoded setpoint value is received. After that value is received, it will turn 90 degrees and move in the second axis and stop after it determines the highest setpoint value.*



3) *Spiral: This algorithm was developed but was not able to be tested in the laboratory setting. This algorithm will move forward until the setpoint value is reached. Afterwards, the system*

will move in a shape like a spiral and gauge all the values in the area and stop at the highest one.



Sensor

The magnetic field sensor we used was the Allegro UGN3503. This sensor accurately tracks extremely small changes in magnetic flux density. It operates within 4-6 VDC, making it convenient to use with the microcontroller of our design. It can also detect magnetic fields with values between -90 and 90 milliTeslas.

Misalignment Percentage

Our algorithm also calculates the percent misalignment between the transmitter and receiving coil. We used a predetermined magnetic field value as a reference and measured different values as the transmitter coil approached the receiving coil. The calculation was based on the following equation to display the misalignment:

$$\frac{|B_{field} - B_{max}|}{B_{max}} * 100$$

$B_{field} = Measured\ magnetic\ field, mT$

$B_{max} = Predetermined\ magnetic\ field, mT$

Algorithm Flowcharts

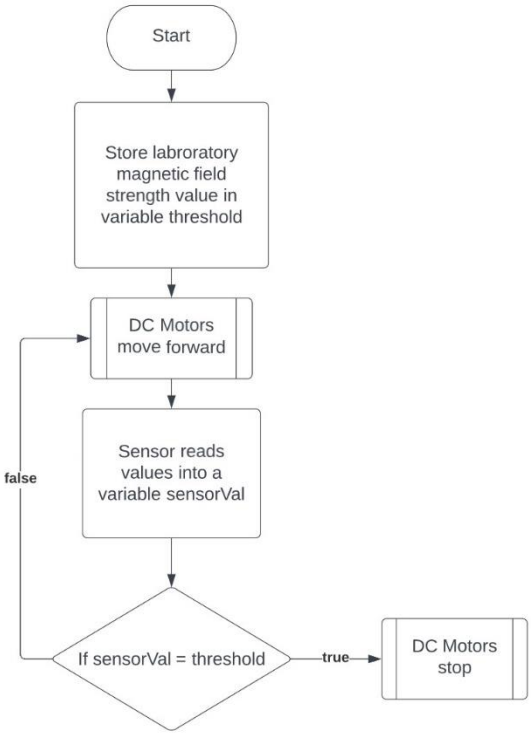


Fig. 11. Single-Axis Alignment Flowchart with threshold

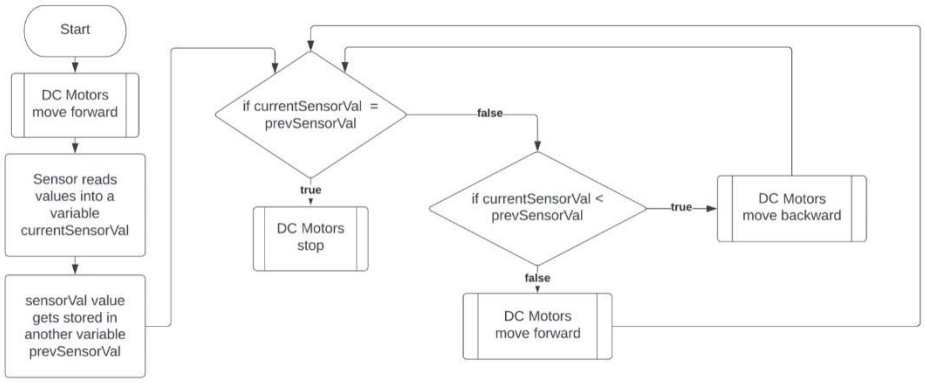


Fig. 12. Single-Axis Alignment with no laboratory measured threshold value

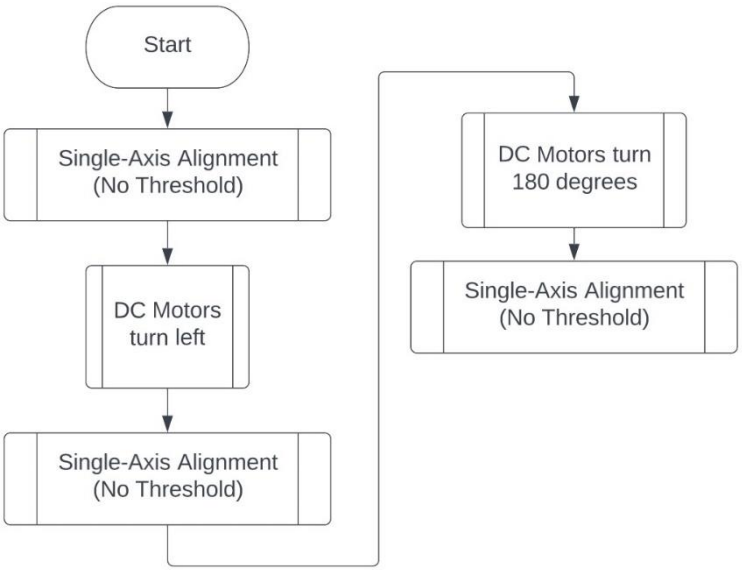


Fig. 13. Consecutive Single-Axis Alignment

Summary

The charging plate and receiver are partially aligned using the EV wireless charging alignment system using a single-axis movement technique. The charging plate is pushed in one way until it recognizes the charging receiver for the car, at which point it reverses direction and moves till it is partially aligned. The misalignment caused by this technique between the charging plate and the receiver can be different based on the receiver's size and shape as well as the location of the car.

IV. RESULTS

This section will present the findings of the different tests conducted. This section will also weigh the results to the project's objectives and analyze the accuracy of the alignment system. The single axis alignment algorithm used in our tests used a predetermined magnetic field value which was used to determine where the alignment system was located. We will also display the misalignment via the microcontroller.

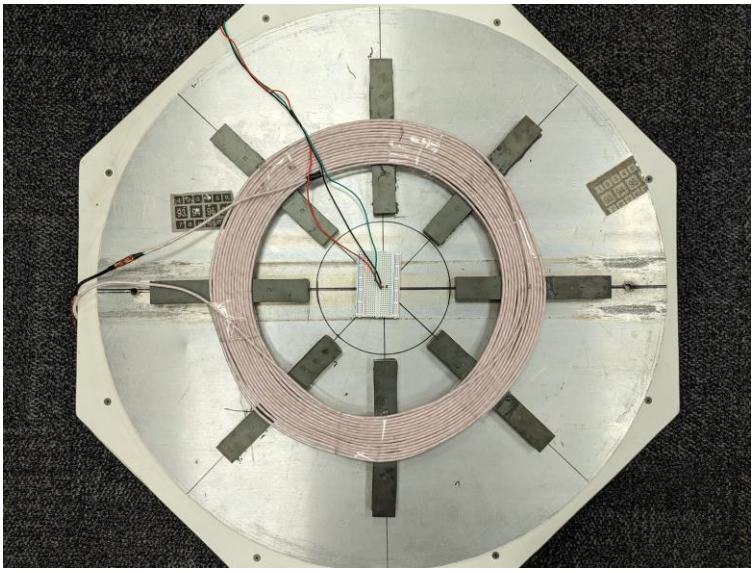


Fig. 14. Alignment Structure Top View

Figure 14 shows the hall effect sensor at the center of our alignment structure. It is structured this way due to the nature of the generated magnetic field. The magnetic field magnitude is largest at the center. Our alignment structure mirrors this by placing the magnetic field sensor at the center. Once the sensor measures the largest value it will be directly underneath the receiving coil.

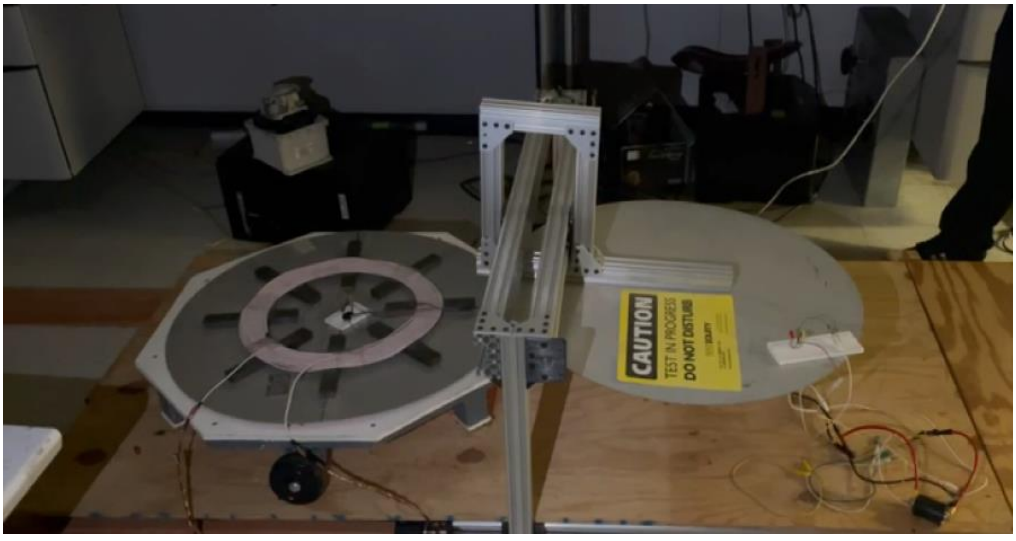


Fig. 15. Laboratory Alignment Path

Figure 15 shows the platform and path of our alignment structure. Due to the fixed mounted receiving coil, testing could only be performed with a single axis alignment algorithm.

Fig. 16. Proper Alignment

Figure 16 shows the alignment structure directly underneath the receiving coil with the transmitting coil transferring power. We used LED lightbulbs to display the wireless transfer of power. We used a DC input source of 30V and generated a high frequency current through our mentor's high frequency converter.

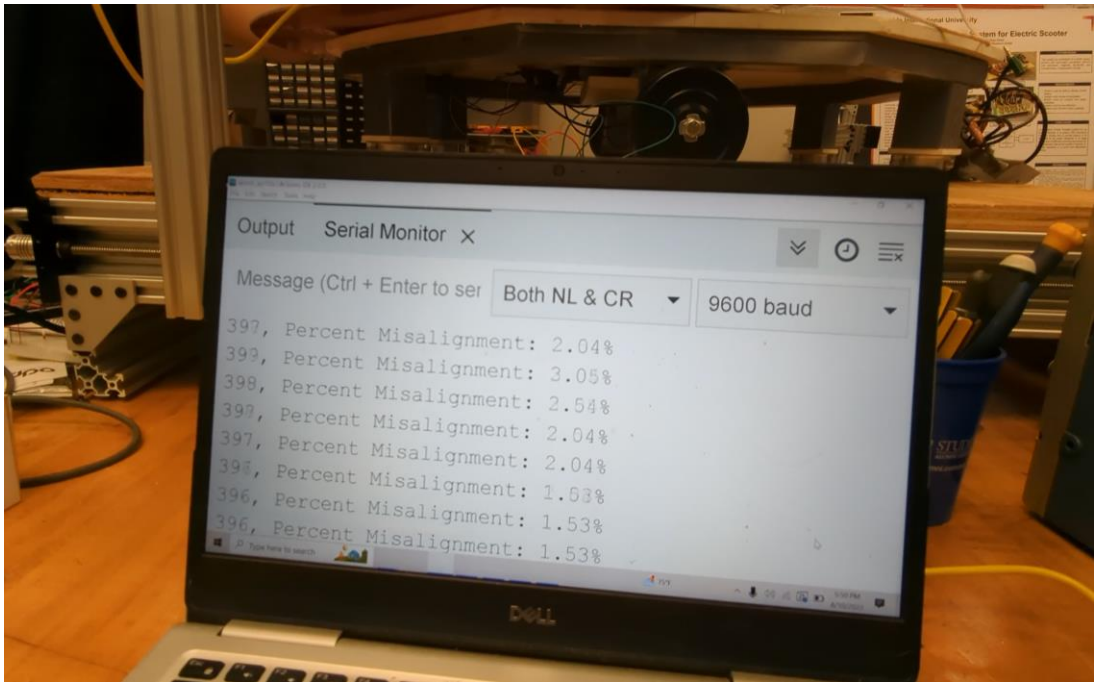


Fig. 17. Misalignment Readings

Figure 17 displays the percentage of misalignment between the receiving and transmitting coil based on our predetermined magnetic field value. The misalignment value decreases as as the transmitter approaches the receiver.

In conclusion, the findings presented in this study demonstrate the effectiveness and accuracy of the single axis alignment algorithm used in our tests. The predetermined magnetic field value used to determine the location of the alignment system allowed for precise measurements of misalignment, which were clearly displayed via the microcontroller. The results of the tests conducted were carefully weighed against the objectives of the project, and it was found that the alignment system successfully met the desired performance criteria. These findings have important implications for the use of similar alignment systems in various applications and industries. Further research could explore the use of different algorithms and methods to enhance the accuracy and efficiency of alignment systems. Overall, this study provides valuable insights into the practical applications of single axis alignment systems and highlights the potential benefits of investing in further research in this area.

V. FUTURE IMPROVEMENTS

The Electric Vehicle Wireless Charging Alignment System is a promising technology that uses magnetic induction to wirelessly charge electric vehicles. However, there are several ways that the project could be improved in the future.

Increasing charging efficiency:

Enhancing the system's charging efficiency is one of the main approaches to making the Electric Vehicle Wireless Charging Alignment System project better. To make sure that the magnetic field is strong enough to charge the EV, the system now depends on the alignment of the transmitting and receiving coils. However, there can be energy losses when charging because the alignment procedure is not always ideal. Advanced algorithms and control systems could be used to optimize the alignment process and guarantee that the transmitting coil is always placed directly underneath the receiving coil. Additionally, to reduce energy losses during charging, materials with higher magnetic permeability and lower electrical resistance could be employed. The system might charge EVs more quickly and consistently by improving charging efficiency, making it even more practical for users.

Extending the range:

Extending the wireless charging range would enable EVs to be charged while being driven or parked in a larger region, which is another approach to enhance the system. Currently, the system's range is constrained, and EVs may only charge when they are parked immediately above the transmitting coil. The transmitter's power output could be raised to expand the range, enabling the magnetic field to cover more ground. The magnetic field sensors' sensitivity and precision might also be increased, which would make it easier for the sending coil to detect and align with the receiving coil fast and accurately. The method might increase the range of EVs, increasing user convenience and accessibility while also removing the need for charging stations.

Enhancing safety features:

Any charging system must prioritize safety, and the Electric Vehicle Wireless Charging Alignment System is no different. Safety safeguards might be improved to guarantee that the system is secure for users and does not provide a risk of electrical shock or electromagnetic interference. To identify any alignment issues or malfunctions, for instance, additional redundant sensors might be installed. Additionally, automatic power cutoffs could be provided to stop overcharging and overheating. To ensure that the system shuts down in the event of a power loss or other emergency, fail-safe methods could also be incorporated. The system may become more dependable and durable and win over more users and investors by strengthening its safety features.

Expanding compatibility:

The Electric car Wireless Charging Alignment System does not currently handle varied car sizes, shapes, or power requirements. It is only compatible with specific types of EVs. Compatibility could be increased to make the system more usable and practical for a larger range of users. Designing flexible transmitting and receiving coils that can adapt to various vehicle profiles could

help with this. The system might also be made modular, enabling users to alter it to suit their own requirements. The system might become even more adaptable and user-friendly by increasing compatibility, which could accelerate the adoption of EVs.

Improving scalability and affordability:

Finally, the system's scalability and cost need to be enhanced to attract users and investors. This could be accomplished by optimizing the manufacturing process, saving money on materials, and raising output levels. Collaboration with EV producers and infrastructure providers may also help expand the technology's market and improve its economic viability. By making scaling easier

VI. RESOURCES

Video Links:

Introduction Video-https://youtu.be/7djVCnZGNiw
Concept Design Video-https://youtu.be/yTgMdjMaAW0

Current Arduino Code Used:

```
 // Imports
#include <stdlib.h>

// Pin Declarations
const uint8_t LED_OUTPUT_PIN = 34;
const int HALLEFFECT_INPUT_PIN = A8;

// Values Depend on Magnet
const uint16_t LOWER_BOUND_MAX_SENSOR_VALUE = 390;
const uint16_t MAX_SENSOR_VALUE = 405;
const uint16_t TURN_DURATION_90_DEG = 10250; // 10000;
const uint16_t TURN_DURATION_180_DEG = TURN_DURATION_90_DEG * 2; // 20500;
// Motors
// Inputs are for wheel spin direction
// Enable for speed

// Motor A:
int enA = 9;
int in1 = 8;
int in2 = 7;

// Motor B
int enB = 3;
int in3 = 5;
int in4 = 4;

// Function Declarations
void runForwardFor();
void runBackwardsFor(int milliseconds);
void turnLeft(int turnDurationMS);
void turnRight(int turnDurationMS);
double calculateMisalignment(int actual, int maxVal);
//double calculatePercentages(int actual, int maxVal);
void runStop();
void runStopFor();
//void runSpiral();
//void singleAxisAlignment();
//double calibrateMidValue(int sampleSize);

// Variable Declarations
int hallEffectVal; // Current magnetic field value
int lasthallEffectVal; // Previous magnetic field value
double percentMisalignment;
// Spiral Function Variable: Requires Tuning
int tourtime = 1000; // circle tourtime. it is about intersection areas
between each spiral. it should be less than circle time
int increasetime = 10; // time for next tour, larger spiral needs more time
int numberofspiral = 10; // number of circle
```

```
int minspeed = 5; // initial speed for spiral.
int maxspeed = 200;// max speed
int ledVal;


void setup()
{

 // redundant
 int lasthallEffectVal;
 int hallEffectVal;
 double percentMisalignment;

// Set all the motor control pins to outputs
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

   // put your setup code here, to run once: <-motors
  pinMode(LED_BUILTIN, OUTPUT);
 /* Lmotor.setSpeed(255);
  Rmotor.setSpeed(255);*/

  //sensor
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  pinMode(HALLEFFECT_INPUT_PIN, INPUT);
  Serial.begin(9600);

}

void loop()
{
  lasthallEffectVal = hallEffectVal;
  hallEffectVal = analogRead(HALLEFFECT_INPUT_PIN);

  // Remap the value for output. In my experiments, the minimum input
  // value for when the north part of the magnet was directly in front of
  // the Hall effect sensor was ~22
  //int ledVal = map(hallEffectVal, 12.5, 1023, 0, 255);

   //double averageMidvalue = calibrateMidValue(50); //
  // To middle
//  if (hallEffectVal >= LOWER_BOUND_MAX_SENSOR_VALUE ){
//    runStopFor(100000);
//    //runStop();
//
//
//  } else {
//    runForwardFor();
////    turnRight(TURN_DURATION_90_DEG);
////    runStopFor(5000);
////    turnLeft(TURN_DURATION_180_DEG);
////    runStopFor(5000);
//  }
```

```
  percentMisalignment = calculateMisalignment(hallEffectVal,
MAX_SENSOR_VALUE);

  // Print the raw photocell value and the converted led value (e,g., for
Serial
  // Console and Serial Plotter)
 //Serial.print(averageMidvalue);
 //Serial.print(",");-
 Serial.print(hallEffectVal);
 Serial.print(", ");
 Serial.print("Percent Misalignment: ");
 Serial.print(percentMisalignment);
 Serial.println("%");
// Serial.print("Percent Misalignment: ");
// Serial.print(percentArr[0]);
// Serial.print(", ");
// Serial.print("Percent Alignment: ");
// Serial.println(percentArr[1]);
 //Serial.println(lasthallEffectVal);

  // Write out the LED value.
  analogWrite(LED_OUTPUT_PIN, ledVal);
  Serial.println(ledVal);

  delay(20); // read at 50Hz
}

double calibrateMidValue(int sampleSize)
{
  int idleValueSample[sampleSize];
  int magneticFieldValueSum = 0;
  double averageMagneticFieldValues;

  for (int i = 0; i < sampleSize; i++){
    idleValueSample[i] = hallEffectVal;
    magneticFieldValueSum += hallEffectVal;
  }

  averageMagneticFieldValues = magneticFieldValueSum / sampleSize;
  return averageMagneticFieldValues;

}

void singleAxisAlignment()
{
   if (hallEffectVal >= LOWER_BOUND_MAX_SENSOR_VALUE ){
    //runStopFor(5000);
    runStop();


  } else {
    runForwardFor();
//    turnRight(TURN_DURATION_90_DEG);
//    runStopFor(5000);
//    turnLeft(TURN_DURATION_180_DEG);
//    runStopFor(5000);
  }
```

```
 Serial.print(hallEffectVal);
 Serial.print(", ");
 Serial.print("Percent Misalignment: ");
 Serial.print(percentMisalignment);
 Serial.println("%");
// Serial.print(",");
// Serial.print(hallEffectVal);
// Serial.print(",");
// Serial.println(lasthallEffectVal);
}



void runForwardFor()
{
  // Set Motor A direction to spin Forward
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);

  // Set Motor A speed (0~255)
  analogWrite(enA, 255);

  // Set Motor B direction to spin Forward
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);

  // Set Motor A speed (0~255)
  analogWrite(enB, 255);


}

void runBackwardsFor(int milliseconds)
{
  // Set Motor A direction to spin Forward
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);

  // Set Motor A speed (0~255)
  analogWrite(enA, 255);

  // Set Motor B direction to spin Forward
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

  // Set Motor A speed (0~255)
  analogWrite(enB, 255);

  delay(milliseconds);
}

void runStopFor(int milliseconds)
{
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  delay(milliseconds);
```

```
}

void runStop()
{
  analogWrite(enA, 0);
  analogWrite(enB, 0);
}

double calculateMisalignment(int actual, int maxVal)
{
  // Works ========================================
  //double percentMisalignment = (abs(actual - maxVal) / maxVal) * 100;
//  double absVal = abs(actual - maxVal);
//  double divi = absVal / maxVal;
//  double percent = divi * 100;
//
//  return percent;
//  //return percentMisalignment;
// Works ========================================
// map(value, fromLow, fromHigh, toLow, toHigh)
  double mappedVal = map(actual, 200, 405, 0, 405);
  double percentMisalignment = (abs(mappedVal - maxVal) / maxVal) * 100;
  double absVal = abs(mappedVal - maxVal);
  double divi = absVal / maxVal;
  double percent = divi * 100;

  return percent;

}

void turnLeft(int turnDurationMS)// change
{ // Motor A: Left; Motor B: Right

  // Set Motor A (LEFT Motor) direction to spin Backward
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);

  // Set Motor A speed (0~255)
  analogWrite(enA, 255);

  // Set Motor B (RIGHT Motor) direction to spin Forward
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);

  // Set Motor B speed (0~255)
  analogWrite(enB, 255);

  delay(turnDurationMS); // how long it spins, optimize value


  analogWrite(enA, 0);
  analogWrite(enB, 0);
}

void turnRight(int turnDurationMS)// change
{ // Motor A: Left; Motor B: Right
  // Set Motor A (LEFT Motor) direction to spin Forward
```

```
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);

  // Set Motor A speed (0~255)
  analogWrite(enA, 255);

  // Set Motor B (RIGHT Motor) direction to spin Backward
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

  // Set Motor B speed (0~255)
  analogWrite(enB, 255);

  delay(turnDurationMS); // how long it spins, optimize value


  analogWrite(enA, 0);
  analogWrite(enB, 0);
}

double calculatePercentages(int actual, int maxVal)
{
//  // Need the actual value
//    // 390
//  double percentages[2]; // [misalignmet, alignment], // array may need to
be a global var
//
//  // Find the Percent
//    // |390-400| / 400
//    // 10 / 400 = 0.025 * 100 = 2.5%
//    // 100 - 2.5
//  double percentMisalignment = (abs(actual - maxVal) / maxVal) * 100;
//  percentages[0] = percentMisalignment;
//
//  double percentAlignment = 100 - percentMisalignment;
//  percentages[1] = percentAlignment;
//
//  return percentages;
}
```