AUTOMATIC POSITIONING SYSTEM FOR INDUCTIVE WIRELESS CHARGING DEVICES AND APPLICATION TO MOBILE ROBOT

A Thesis

by

IVAN CORTES

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2017

Major Subject: Mechanical Engineering

ABSTRACT

Inductive power transfer (IPT) remains one of the most common ways to achieve wireless power transfer (WPT), operating on the same electromagnetic principle as electrical transformers but with an air core. IPT has recently been implemented in wireless charging of consumer products such as smartphones and electric vehicles. However, one major challenge with using IPT remains ensuring precise alignment between the transmitting and receiving coils so that maximum power transfer can take place. In literature, much of the focus is on improving the electrical circuits or IPT coil geometries to allow a greater transmission range. Nevertheless, most IPT products today rely on precise alignment for efficient power transfer.

In this thesis, the use of sensing coils to detect and correct lateral misalignments in a typical IPT system is modeled and tested. The sensing coils exploit magnetic-field symmetry to give a nonlinear measure of misalignment direction and magnitude. To test this idea, three experiments are performed: 1) measure the voltage of experimental sensing coils for various lateral misalignment distances, 2) implement closed-loop control and measure performance for an experimental two-dimensional (2D) automatic IPT alignment mechanism, and 3) test automatic IPT alignment on a plausible mobile robot wireless charging scenario. The experimental sensing coils give a misalignment sensing resolution of 1 mm or less in two lateral directions, allowing automatic alignment control in real time with a maximum lateral positioning error of less than $\sqrt{2}$ mm. This precise alignment allows for efficient power transfer to occur. When implemented on the mobile robot platform, the automatic positioning system gives similar results, allowing the robot to position itself above a wireless charger precisely—a task the mobile robot cannot accomplish using its navigation camera alone. The results of this experiment give confidence that similar sensing coils can be used to reduce lateral misalignments in scaled IPT systems, such as electric-vehicle wireless chargers.

DEDICATION

I dedicate this thesis to my brothers, to my sister, and to my parents who have supported me in everything I do since I can remember. Thank you, Eric, Hiram, Dario, Leslie, mom, and dad.

ACKNOWLEDGMENTS

I thank my advisor, Dr. Kim, for his guidance and support throughout the course of this research and my pursuit of the master's degree. I also thank Dr. Rathinam and Dr. Amato for serving on my thesis committee.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professors Won-Jong Kim and Sivakumar Rathinam of the Department of Mechanical Engineering and Professor Nancy Amato of the Department of Computer Science and Engineering.

Several commercially-available products detailed in this thesis (Arduino, Raspberry Pi, Qi wireless charger, and others) were used for experimentation. Open-source software and library packages were also used. All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work was made possible in part by the National Science Foundation through a fellowship award to Ivan Cortes. Award No. HRD-1502335. "Texas A&M University System Louis Stokes Alliance for Minority Participation (TAMUS LSAMP) Bridge to the Doctorate (BTD) Cohort XI (2015-2017) Program."

NOMENCLATURE

AC	Alternating current
ADC	Analog-to-digital conversion
DC	Direct current
DOF	Degrees of freedom
FOV	Field of view
GPIO	General-purpose input/output
HSV	Hue, saturation, value
IPT	Inductive power transfer
MIMO	Multiple-input multiple-output
MQS	Magnetoquasistatic
P	Proportional
PWM	Pulse-width-modulation
RGB	Red, green, blue
SISO	Single-input single-output
WPT	Wireless power transfer
1D	1-dimensional
2D	2-dimensional
3D	3-dimensional

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Most electronic devices today rely on wired connections for battery charging or direct power. However, wires require secure physical connections that limit freedom of movement. Solutions for this include using flexible connections like brushes, slip rings, or liquid conductors. Many of these connections remain susceptible to friction wear, contact resistance, material fatigue, or environmental effects, such as oxidation or contamination.

An alternative to wired connections is wireless power transfer (WPT). WPT refers to any method of power transfer which does not use wires, and may include using light, radio frequency (RF), acoustics, electric fields, or others. However, one of the most common methods for WPT is inductive power transfer (IPT). As the name suggests, this method relies on an inductive link between coils to transfer energy.

1.1 Origin of inductive power transfer

Some of the first experiments with IPT were conducted by Tesla in the late 1800's [1]. His experiments included highly-resonant transformer circuits and inductive coils that could power electronic devices though the air. Earlier in the same century, Faraday and Henry had discovered electromagnetic induction using coils of wire and various transmission media [2]. In 1885, the first electrical transformer, which operated using IPT, was designed by Stanley [3]. The magnetic induction principle gave way to the invention of various electromagnetic machines, such as electric generators and motors. Some of the first consumer products to use IPT for wireless charging included electric toothbrushes, and the creation of the Qi wireless charging standard in 2008 prompted widespread adoption of the technology for many other consumer products [4].

Today, IPT is still the preferred method for short and mid-range WPT since these ranges allow for an inductive link between coils in proximity of each other even through the air. In a typical alternating-current (AC) transformer, the inductive link between the primary and secondary coil is strengthened by using an iron core and placing the coils very close to each other, allowing for transfer efficiencies easily exceeding 95%.

Other methods of wireless power transfer have not been as widely implemented due to some limitations. Capacitive WPT generally requires large voltages or large surface areas. Light, RF, and acoustic-based transfer are susceptible to attenuation by foreign objects. Nevertheless, there is some notable work being done in these areas [5–8].

1.2 Inductive wireless power transfer

A simplified version of a typical IPT system is shown in Figure 1. The main components of this setup are an AC power source, primary coil, secondary coil, and rectifier circuit. Two additional components, compensators, are shown in a lighter shade. Compensator circuits are not required, but are essential for increasing transfer efficiency. The compensators may include a variety of resonance-shaping schemes or advanced functions for controlling the power to the primary coil. The space between the primary and secondary coil in Figure 1 may be filled with different media. In a general WPT scenario for wireless charging, this space is filled by ambient air.

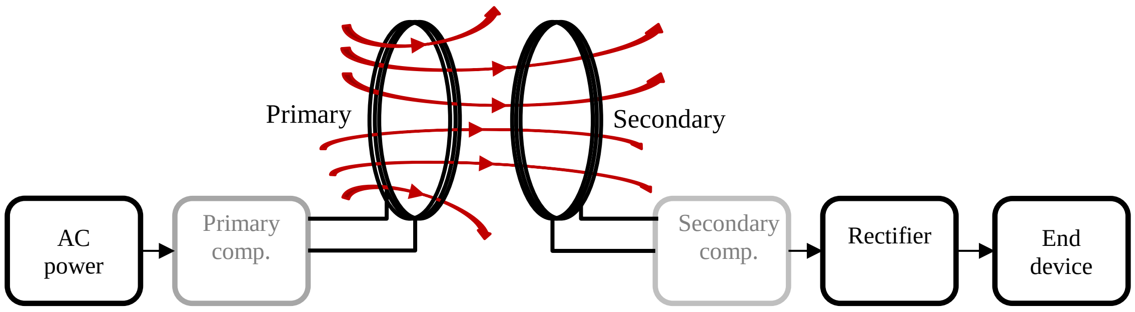


Figure 1. Simplified diagram of an inductive power transfer (IPT) system. The curved arrows represent magnetic field lines created by the primary coil.

The following is a brief explanation for the role of each component presented in Figure 1:

- AC power: This can be a voltage or a current source. The working frequency may vary from kilohertz to megahertz depending on the application. If powered by a DC source, this component will include an inverter.
- Primary coil: This coil is made with one or more turns of insulated wire or conductive tube and creates a concentrated magnetic field which alternates at the same frequency as the AC power source.
- Secondary coil: This coil need not have the same number of loops or shape as the primary coil. Some of the time-varying magnetic flux from the primary coil penetrates the secondary coil. By Faraday's law of induction, a voltage is induced in the secondary coil due to the varying magnetic flux.

- Rectifier: The voltage induced in the secondary coil will alternate at the same frequency as the AC power source. The rectifier converts voltage to DC, if necessary, to power an end device.
- Compensators: These represent a variety of specialized circuits that improve transfer efficiency. For example, they may shape the resonance frequency of the coils by introducing series or parallel capacitors. They may also introduce smart control to the system, modulating the output frequency or amplitude of the primary coil depending on the demand or presence of a secondary coil.

Designs of IPT systems vary in selection of the components of Figure 1. Variations may include different coil geometries, power source designs, or compensation schemes. The compensation scheme used depends on the application. Resonance-based IPT has been shown to improve efficiency especially in mid-range power transfer applications [9]. Common resonance schemes include series-series, parallel-parallel, series-parallel, or parallel-series compensation [10, 11]. These scheme names refer to the placement of resonance capacitors on the primary and secondary coils, respectively. Many wireless chargers available on the market today adhere to standards that specify the configuration of IPT components. One of the most popular standards is the Qi standard. This published standard has been adopted by many companies and has been implemented in wireless chargers for products such as smartphones and tables. The most recent Qi standard specification allows up to 15 W of power transfer and operates at a frequency from about 100 kHz to 200 kHz [4].

1.3 Limitations of inductive wireless charging

Figure 2 shows normalized magnetic flux density vectors created by circular loop of wire, calculated using the Biot-Savart law (1). The flux density vectors are scaled for visibility and to show relative magnitudes. The loop of wire in Figure 2 represents a primary coil in an IPT system. Some of the magnetic flux density vectors penetrate a secondary coil in proximity—the rest are fringed into space.

$$\boldsymbol{B} = \frac{\mu_0 N I}{4\pi} \int \frac{d\boldsymbol{l} \times \boldsymbol{r}}{|\boldsymbol{r}|^3} \ , \tag{1}$$

where

$\boldsymbol{B}$: magnetic flux density [T]	N: number of coil loops
$d\boldsymbol{l}$: wire segment [m]	I: current through wire [A]
$\boldsymbol{r}$: vector from wire segment to point of interest	$\mu_0 = 4\pi\times10^{-7}$[H/m]

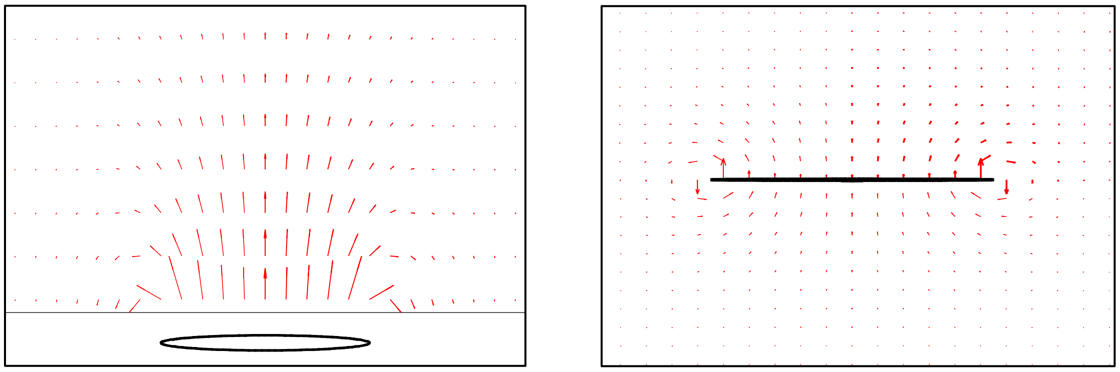


Figure 2. Magnetic flux density vectors due to a current loop. Magnitudes shown are scaled for visibility

Figure 2 suggests that to experience the greatest energy transfer, the secondary coil should be close, parallel, and concentric with the primary coil. The main issue with IPT lies here since, to achieve appreciable energy transfer, the secondary coil must remain aligned with the primary coil. Outside of the region directly above the primary coil, the magnetic flux density is too weak to induce a voltage in the secondary coil. Previous work quantifies the loss of efficiency for different misalignment conditions [12, 13].

Previous research in IPT has largely focused on increasing the operating range for efficient power transfer. Some ways to increase the operating range of IPT are to increase the operating frequency, input power, or dimension of the primary coil. Each of these would increase the magnetic flux magnitude or its time rate of change in the space around the primary coil. There are other effects to consider, such as resistive losses in the coils, skin effect, interaction with foreign objects, and power supply limitations. Much work has been done in deciding optimal coil characteristics for IPT [14, 15].

Another way to increase working range is to increase the number of coils used. For instance, there can be an array of primary coils instead of just one [16–19]. Multiple secondary coils in novel arrangements have also been used to capture more magnetic flux when there is a misalignment [20].

Still, another way to increase the IPT operating range is to exploit coil resonance. As mentioned before, this approach is called resonance-based IPT and involves tuning the primary and secondary coils to the same resonance frequency. Most commercially-available wireless chargers use some form of this. In previous research, there are even examples of using intermediary resonance coils between the primary and secondary to effectively increase the total range of power transfer [21].

Commercially, the most common solution for misalignment intolerance is to ensure good alignment between the primary and secondary coils and to use a tuned resonance compensation scheme of some kind. The popular Qi standard, for example, uses series-series compensation for coil resonance, but also relies on close alignment between the primary and secondary coil, with a typical misalignment tolerance of less than 5 mm. Still, the misalignment tolerance is only a few centimeters for 5-W chargers using this standard. To help users align their devices on these chargers, manufacturers place visual markers on the charger or construct it so that the device falls into alignment (Figure 3).

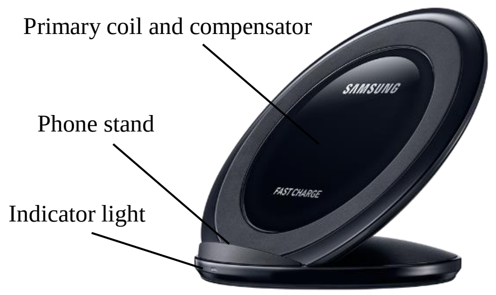


Figure 3. Example of a commercially available Qi wireless phone charger that relies on precise alignment to function efficiently (Samsung model EP-NG930TBUGUS)

1.4 Misalignment detection and coil positioning for IPT systems

One way to maintain alignment between a primary and secondary coil of an IPT system is to use active positioning, where either the primary or the secondary coil is moved in real time to align with the other. There are currently a few examples of active positioning. The Qi standard publication suggests the idea of using a capacitive touch surface or, alternatively, an array of sensing coils to detect the location of the secondary coil. There is also mention of using a positioning mechanism to provide automatic alignment for the IPT system. No design is given for the positioning mechanism, but the standard suggests a positioning resolution of 0.1 mm or better [4]. The standard also presents some guidelines for device detection based on coil impedance and frequency measurements.

One example of active positioning can be found in the online electric vehicle (OLEV) experiments of South Korea [22, 23]. These feature the application of IPT for wireless charging of electric vehicles, where the primary coils are installed in the road itself. The first generation of this experiment used a golf cart as the test platform, which carried an automatic positioning system to keep the secondary coil aligned with the in-road primary coil. Details of the positioning system were not made clear, but it is apparent that a camera sensor was used to detect misalignments. Subsequent generations of this project excluded an alignment mechanism, focusing instead on creating a more favorable magnetic-field distribution to tolerate more misalignment.

In their contribution to vehicle wireless charging research, one group recently used two small sensing coils placed next to the secondary coil on a vehicle to detect misalignment between the vehicle and the in-road wireless power transmitter [24]. These sensing coils were used to measure the difference in magnetic field between the left and right side of the vehicle. A difference indicated that the vehicle was not centered above the in-road primary coil. This information was then used to steer the vehicle and maintain a good alignment with the charger. The steering control was already a function included in the vehicle platform.

Many methods have been used to detect misalignments in IPT systems, some of which were previously mentioned. In electric vehicle charging, lateral misalignment between the ground transmitter and the vehicle-mounted receiver have been detected using magnetic sensing, camera sensors, ultrasonic sensors, and RF sensors [25]. One research group used frequency and phase measurements of the wireless receiver (secondary coil) to determine the misalignment of the electric vehicle over the wireless charger [25]. In this study, additional coils, named auxiliary coils, were added symmetrically to the secondary coil to detect minor lateral misalignments. These measurements could be used to assist the driver in parking the vehicle over the wireless charger for efficient power transfer.

1.5 Applications of inductive wireless charging

With recent advancements in IPT and, especially, the release of wireless charging standards like Qi, an increasing number of wireless charging-enabled products has reached the market. Electric toothbrushes with wireless charging have been available for years. Wireless charging has become a premium feature in many high-end smartphones and tablets. Wireless charging of electric vehicles has been implemented in some public parking spots and is available for private home use as well [26]. Typically, these vehicle-charging systems include a stationary charger and a vehicle parking assistance device to ensure good IPT coil alignment. Misalignment leads to inefficient electric vehicle charging [12, 13]. The position assistance device included with electric vehicle chargers helps the driver park in the correct spot so that the secondary coil can be aligned precisely above the charger. Wireless phone chargers suffer the same problem but phones can easily be adjusted to align with markings on the charger.

IPT has other usages, such as wireless tracking of rodent in a cage [27, 28]. Another application is wireless charging of implanted medical devices [21, 29]. Both applications suffer even more from misalignment since the location and orientation of the secondary coil is more unpredictable. Solutions for the misalignment problem in these cases may include using multiple IPT coils or smart control of charging circuits.

1.6 Contributions of this thesis

The focus of this thesis is to develop a method to sense lateral misalignment in IPT wireless charging systems and to provide a means for correcting this misalignment. To achieve this goal, the following three tasks are pursued:

1. Develop a method, preferably non-intrusive, for detecting lateral misalignment in an IPT charging system.
2. Implement an experimental automatic positioning mechanism that can reduce lateral misalignment to allow efficient power transfer.
3. Demonstrate the use of an automatic positioning mechanism in a mobile robot wireless charging application.

The use of autonomous mobile robots has been proposed for many settings, including cleaning, surveillance, product transport, and other services [30–32]. To maximize productivity, mobile robots can benefit from autonomously recharging themselves at designated charging stations. Power transfer can be accomplished using electrical contacts, but using a misalignment-tolerant wireless charger would reduce contact wear and risk of short circuits, and could allow for completely enclosed circuits.

The mobile robot wireless-charging scenario which is considered in this thesis is depicted in Figure 4. In this scenario, a mobile robot platform is equipped with a secondary coil for IPT. The primary coil (a wireless charger) is located some distance away from the robot and is marked by a colored target. A camera sensor onboard the mobile robot allows the robot to detect the colored target and approach it. However, the camera sensor alone is not precise enough to align the mobile robot with the wireless charger. The IPT misalignment sensor developed in this thesis complements the mobile robot platform by providing the fine misalignment detection needed to ultimately align the mobile robot with the wireless charger—allowing for efficient power transfer to take place.

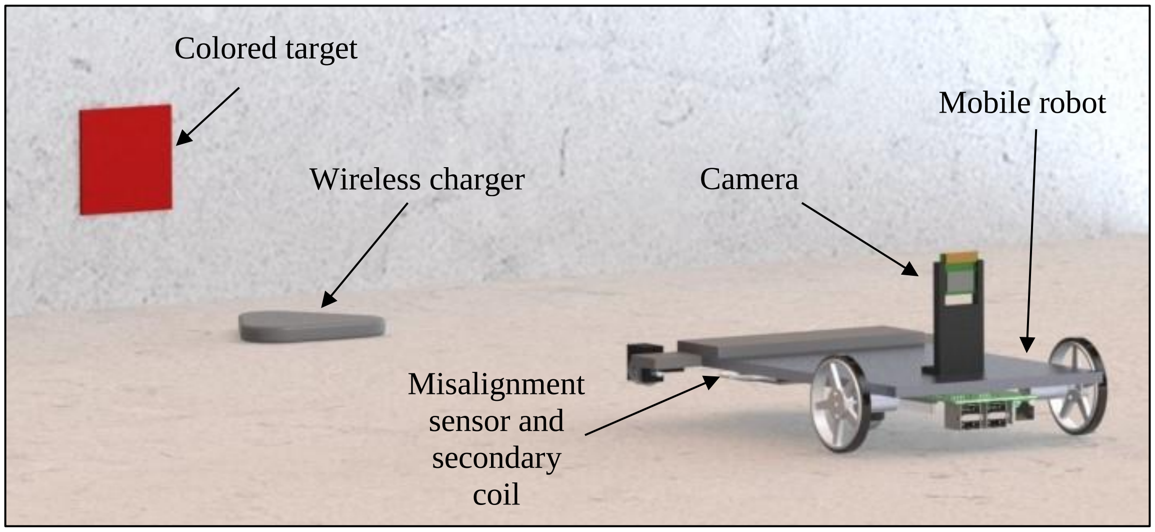


Figure 4. Mobile robot IPT wireless charging scenario

If successful in detecting and automatically correcting lateral misalignments, the results of this thesis can be applied to similar IPT systems, such as wireless charging of electric vehicles, mobile robots, and autonomous vehicles.

1.7 Methods

To accomplish the three tasks presented in Section 1.6, a hypothesis is formed for each task. The three hypotheses are:

1. Sensing coils can be used to sense lateral misalignment direction and magnitude in an IPT system by detecting imbalances in the magnetic field around the secondary coil.
2. With sensing coil misalignment detection, closed-loop control of a positioning system can be used to automatically align an IPT system precisely.
3. An automatic positioning system can aid a mobile robot in aligning with a wireless charger as long as the robot can approach the wireless charger.

The method for testing each of these hypotheses follows the steps of modeling and simulation, experimentation, and discussion.

First, each of the systems introduced in the thesis tasks is analyzed using the appropriate physical laws, approximations, and simulations to predict the behavior of the real systems. For example, the IPT system coils are modeled as circular coils with no resistive losses and ideal current sources. The outcome of this analysis is plots of expected results for the real systems.

Second, appropriate hardware and software are selected for experiments designed to support the analysis results. The experimental setups are described and physical variables are established. Any limitations of using the selected hardware are also identified.

Finally, experimental results for the experiments are discussed and their significance is highlighted in the context of the hypotheses. Whenever appropriate, the experimental results are compared with analysis results. Final conclusions summarize the thesis findings and suggest future work that can be done in automatic alignment of IPT devices.

CHAPTER II

ANALYSIS

The following analysis provides a basic understanding of an IPT system and mobile robot platform. The analysis aids in the design of an IPT lateral misalignment sensor, a misalignment reducing mechanism, and its implementation on a mobile robot.

2.1 Analysis of IPT system

In an IPT system, a current is provided by an AC source to create a magnetic field around the primary coil (Figure 1). Even at high frequencies—typical of IPT systems—the primary coil circuit is tuned so that significant current flows and a magnetic field exists. Ideally, there are no significant sources of electric field. Therefore, the magnetic field effects dominate the system and it can be treated as a magnetoquasistatic (MQS) system. Under this assumption, Faraday's law (2) can be used to calculate the induced voltage in a secondary coil that is placed within the magnetic field of the primary coil.

$$v_{induced} = -\frac{d}{dt}\oint \boldsymbol{B} \cdot d\boldsymbol{a} \qquad (2)$$

Equation (2) states that the secondary coil will experience an induced voltage equal to the time derivative of the magnetic flux that penetrates the coil. The induced voltage in the secondary coil will alternate at the same frequency as the AC source of the and, after being rectified, the voltage in the secondary coil acts like a direct-current (DC) source that can power an end device. Equation (2) implies that to increase the voltage induced in the secondary coil, the time-varying magnetic field penetrating the secondary coil should be increased. Hence, the mutual inductance between the coils should be maximized through their placement relative to each other to maximize power transfer. The mutual inductance is greatest when the primary and secondary coils are parallel, concentric, and close to each other. The fringing of the magnetic field cannot be ignored and the mutual inductance is a function of five misalignment directions, as shown in Figure 5. Misalignment in any of the five directions in Figure 5 will reduce the mutual inductance and total power transfer of the IPT system.

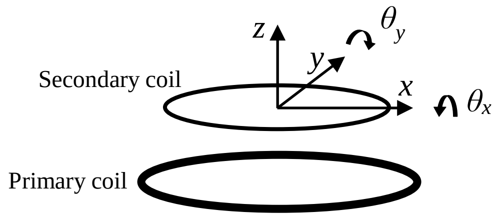


Figure 5. Misalignment directions that decrease power transfer

The AC power source is assumed to provide a sinusoidal current to the primary coil with time frequency ω. From (1), the magnetic flux density ***B*** everywhere will also vary sinusoidally at the same frequency. Substituting a sinusoidal magnetic flux density into (2) gives an induced voltage in the secondary coil given by (3).

$$v_{induced} = \left(n\,\omega \oint B_{\perp} da \right) \sin(\omega t) \qquad (3)$$

The dot product from (2) is realized in (3) by keeping only the component of the magnetic field perpendicular to the secondary coil area ($B_{\perp}$). The number of turns in the secondary coil is *n* and multiplies the induced voltage. With *n* and ω as constants, the amplitude of the induced voltage in (3) and can be used as a measure of the magnetic flux penetrating the secondary coil. The amplitude of the induced voltage (4) can easily be measured.

$$v_{induced,\ amplitude} = n\,\omega \oint B_{\perp} da \qquad (4)$$

All subsequent analysis of the IPT system will be based on (4) and the magnetic flux densities calculated using (1). In this way, the induced voltage in any coil within the primary coil magnetic field can be calculated. In using (4), the time-derivative of the magnetic flux due to movement of the coils is ignored; time-varying magnetic flux due to movement of the coils is negligible compared to the varying flux caused by the high-frequency AC current.

The integral in (4) can be approximated by sampling the magnetic flux density vectors in the secondary coil area. Furthermore, if misalignments of the secondary coil are constrained to the x and y directions of Figure 5, then $B_{\perp}$ becomes B_z—the z component of the magnetic flux density. Constraining motion of the secondary coil to the x and y directions is consistent with many applications of IPT, including the mobile robot wireless charging scenario of Figure 4. For the remainder of this thesis, only misalignments in the x and y direction are considered and collectively called lateral misalignments.

2.2 Misalignment-sensing coils

To correct lateral misalignments in an IPT system, it would be beneficial to know the magnitude and direction in which misalignment occurred. Knowing that misalignment results in decreased power transfer, one might monitor the induced voltage in the secondary coil to detect changes in power transfer. However, changes in power transfer could occur for reasons other than misalignment, such as variations in the power source.

In addition, there are many misalignment directions which would lead to decreased power transfer (see Figure 5). Thus, a different method is proposed in this thesis.

The proposed method is to use additional sensing coils, like those used in [24, 25]. The use of sensing coils is based on the observation that the magnetic field is symmetric about the primary coil *z* axis in Figure 5. Thus, lateral misalignments result in magnetic field imbalances around the secondary coil. These imbalances could be measured using sensing coils that can operate independently of the IPT system as long as the primary coil is powered.

Consider the case where misalignment occurs in the *x* direction of Figure 5. Two small sensing coils are introduced and placed as shown in Figure 6. The sensing coils are attached symmetrically to the secondary coil and move along with it. Just as the secondary coil experiences an induced voltage from the time-varying magnetic field, each sensing coil also experiences and induced voltage (v_L for the left sensing coil and v_R for the right sensing coil). However, due to the symmetric magnetic flux density distribution shown in Figure 2, a different magnetic flux penetrates the sensing coils when there is a lateral *x* misalignment. Thus, *x* misalignment results in different induced voltage magnitudes for the sensing coils. This voltage difference can be measured to detect misalignments.

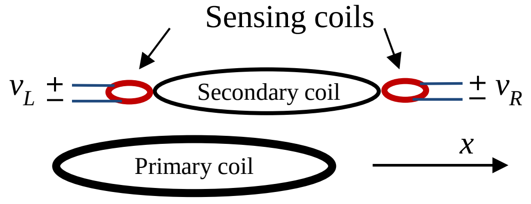
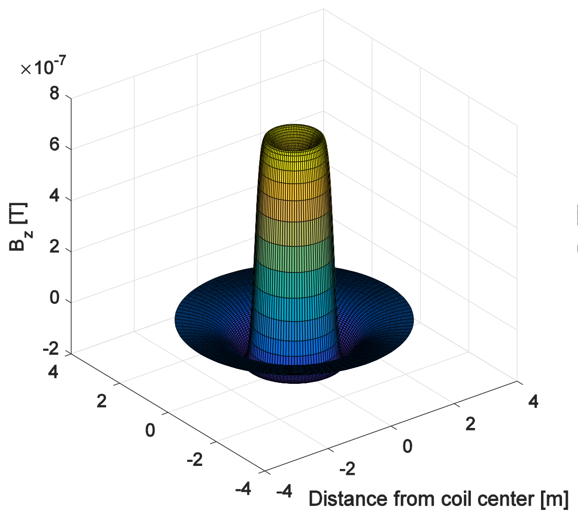


Figure 6. Proposed sensing coils for misalignment detection in one direction

When there is no *x* misalignment, the sensing coils in Figure 6 experience the same induced voltage—again, due to the magnetic field symmetry. Thus, a near-zero induced voltage difference between the sensing coils serves as an indication that the IPT system is in alignment.

The *z* component of the magnetic flux density distribution at two fixed heights from the primary coil are plotted (Figure 7). As previously discussed, only the *z* component of the magnetic flux will induce a voltage in the sensing coils because they remain parallel to the primary coil. The SI unit values of operating current and primary coil radius are used for the magnetic flux calculation of Figure 7 to normalize results. Note that the magnetic flux density distribution is indeed axisymmetric. Furthermore, note that the greatest magnetic flux densities are concentrated above the primary coil area.



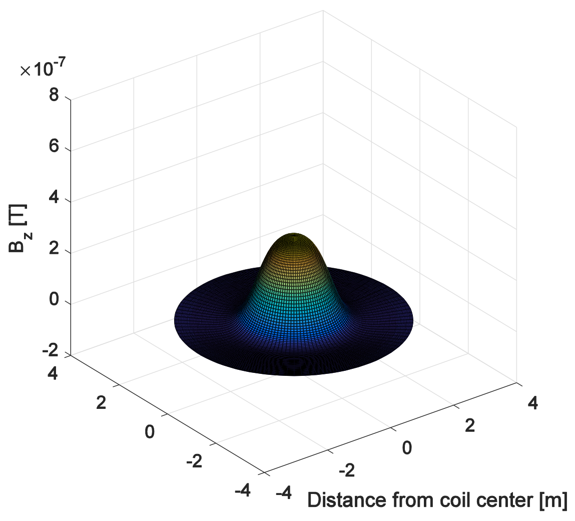


Figure 7. Magnetic flux's *z* component at a height of 0.25 m (left) and 0.75 m (right) due to a single loop of wire carrying a current of 1 A

The axisymmetric magnetic flux distribution shown in Figure 7 means that sensing coils such as those in Figure 6 can be used to detect misalignment in any radial direction. Two more sensing coils are added to detect misalignments in the *y* direction (Figure 8).

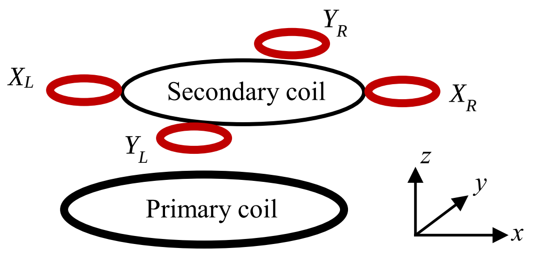


Figure 8. Sensing coil configuration for lateral misalignment sensing

Figure 9 shows the radial dependence of B_z for various heights. The zero-crossing locations at each height are also included.

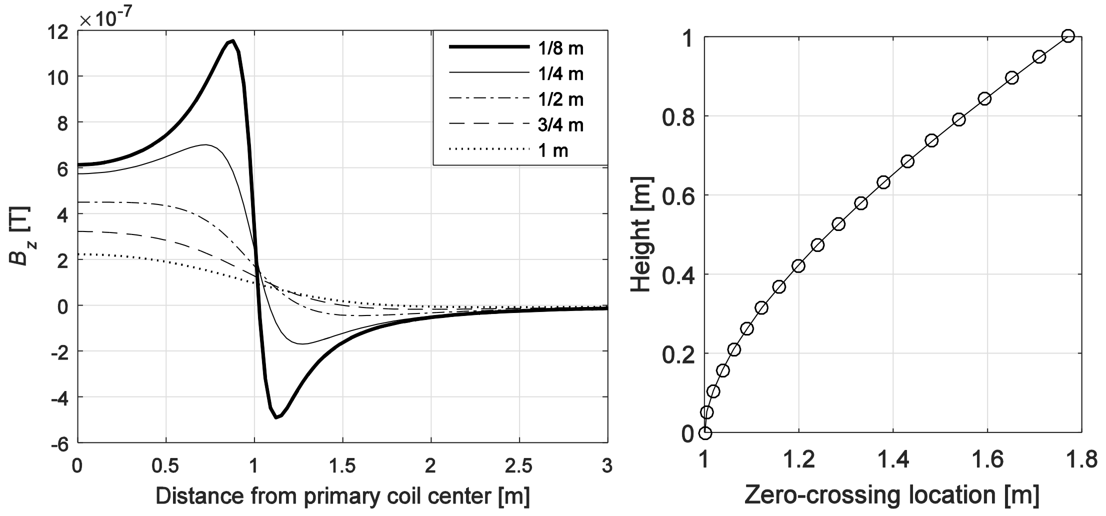


Figure 9. Magnetic flux density's *z* component at various heights (left) and the zero-crossing location at each height (right)

The curves in Figure 9 can be used to predict the magnitude of the induced voltage in sensing coils that exists parallel to the primary coil. For instance, the following predictions can be made:

- At a greater height, the induced voltage in a sensing coil will be smaller because the magnetic flux density is weaker.
- From the zero-crossing plot, it is evident that the height affects how much misalignment the sensing coil can experience before it does not experience a sensing coil voltage.

These predictions are for a single sensing coil. To detect misalignment in an IPT system, the use of two sensing coils is proposed. Figure 10 illustrates how the magnetic flux distribution affects two sensing coils differently.

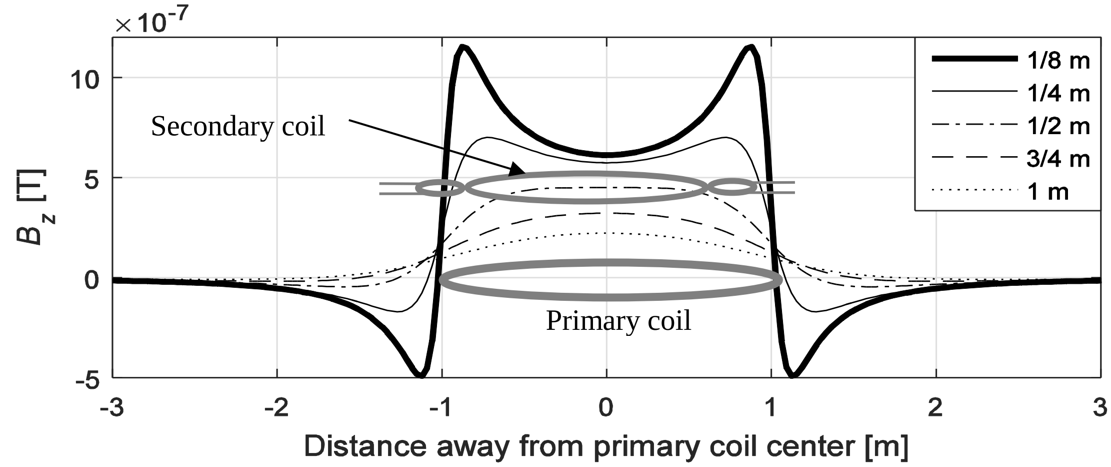


Figure 10. Visualization of magnetic flux penetrating sensing coils as they move with the secondary coil in an IPT system (curves reproduced from Figure 9)

From Figure 10, one can see that the two sensing coils experience a different induced voltage when misalignment occurs. The expectation is that this voltage difference can be measured to detect misalignments in the IPT system.

2.3 Expected results for sensing coils

Using (4) and a discrete approximation of (1), the induced voltage for misalignment sensing coils was calculated for different sensing coil diameters, heights from the primary coil, and misalignment amounts. The calculations were performed in a Matlab script (Appendix A). A custom function uses (1) to calculate the magnetic flux density vectors in the space around a loop of current-carrying wire. The SI unit quantities were used for primary coil radius, operating current, and frequency to obtain normalized results. The normalized results are expected to be useful in predicting the behavior of scaled IPT systems.

Figure 11 shows the calculated voltage difference for dual sensing coils at a height of one 1 m from the primary coil. The results of four different sensing coil diameter sizes are included and the misalignments are constrained to one direction only (the *x* direction in Figure 5). For each sensing coil diameter size, the sensing coils are placed so that their outermost edges are one primary coil diameter apart. This sensing coil placement ensures that both sensing coils are within the strong magnetic flux region when there is no misalignment (see Figure 10). If the sensing coils are placed too far apart, then zero misalignment and small misalignments alike would leave the sensing coils outside of the magnetic flux region and no induced voltage would occur.

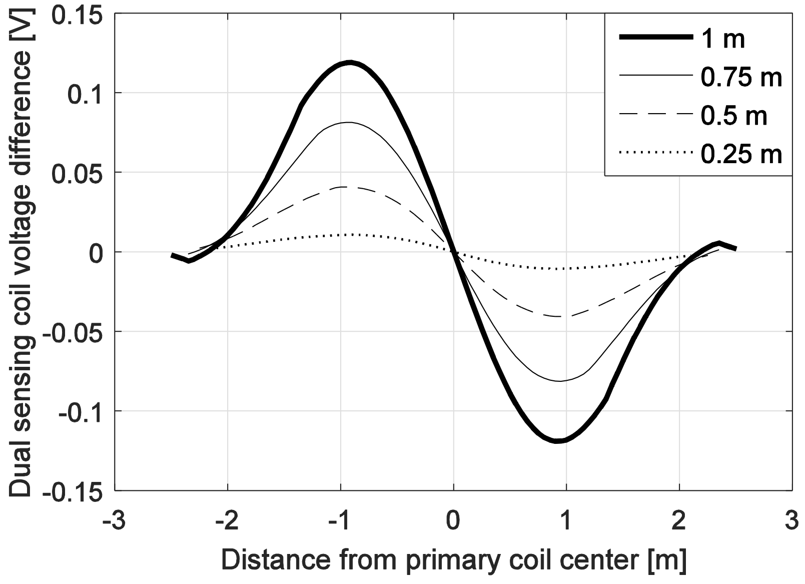


Figure 11. Voltage difference for dual sensing coils versus *x* misalignment for various sensing coil diameters, at 1 m height

In Figure 11, the magnitude of the voltage difference increases with misalignments up to about 1 m in either direction. Beyond 1 m, which is the primary coil radius, the voltage difference magnitude begins to decline. The voltage difference is positive in one direction of misalignment and negative in the other, which suggests that the sensing coil voltages are useful for detecting misalignment direction. However, due to the nonlinear voltage output, the sensing coils are not useful for measuring misalignment amount unless the misalignment is known to be less than one primary coil radius. The result in Figure 11 also suggests that the maximum detectable misalignment is about 2 m, or one primary coil diameter. Finally, note that a larger sensing coil diameter results in larger voltages. Larger voltages could also be achieved by increasing the number of turns in the sensing coils, according to (4).

To see how height from the primary coil affects the voltage difference between the sensing coils during misalignment, the sensing coil diameter is fixed at 1 m and the results for different heights are plotted (Figure 12).

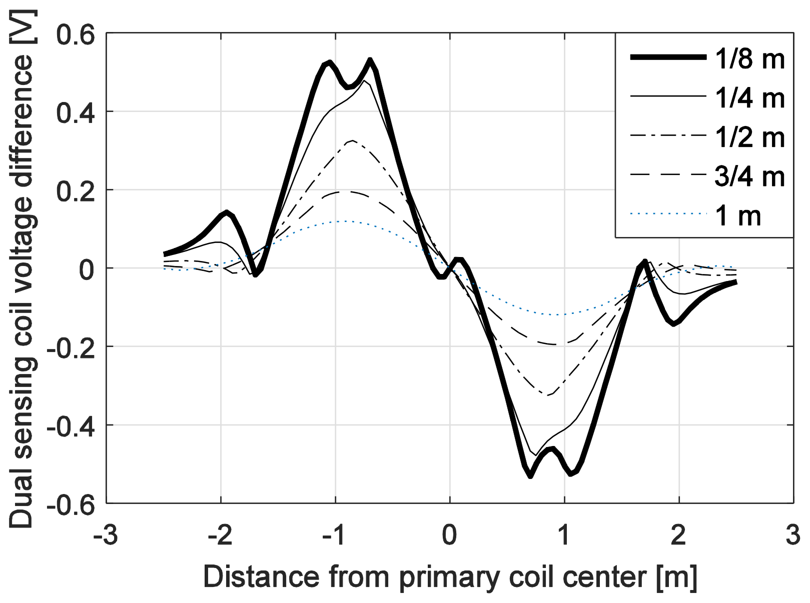


Figure 12. Voltage difference in sensing coils versus *x* misalignment distance at various heights, with sensing coil diameter of 1 meter

In Figure 12, it is apparent that the voltage trend remains the same at most heights, although greater voltages are obtained at lower heights due to a stronger magnetic flux density (see Figure 9). At the lowest plotted height of 1/8 m, however, the trend experiences some anomalies; these manifest as small peaks near zero and one meter misalignment. To further explore these anomalies, the same height of 1/8 m is used with different secondary coil dimeters (Figure 13).

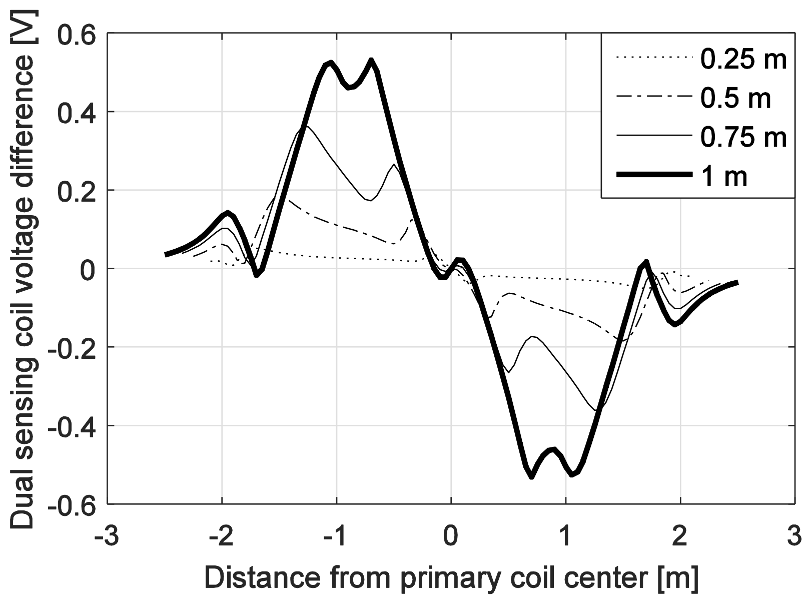


Figure 13. Voltage difference in dual sensing coils versus *x* misalignment for various sensing coil diameters, at 1/8 m height

Figure 13 shows that anomalies (in the form of voltage spikes) occur for many sensing coil diameters when the height is low. Therefore, it is speculated that for dual sensing coils to work as desired, the operating height should be greater than 1/8 m. The voltage spikes at low heights can be explained by the magnetic flux density distribution of Figure 9, where local flux density spikes are also seen at low heights.

Results in Figures 11−13 suggest that, regardless of the height and sensing coil diameter, the misalignment sensing range is about one primary coil diameter. This occurs because the sensing coils measure a noticeable voltage only when they are within (approximately) the area directly above the primary coil radius.

2.4 Dual sensing coils in general

The analysis plots from Figures 9−13 were calculated using the SI unit values for the primary coil radius and operating conditions (frequency and current). Using the unit values means that the results are normalized and can be used as a general representation of other systems. This claim is now shown to be true in simulation.

From (1) and (4), operating current, number of coil windings, and operating frequency are constants of integration. As a result, the flux in a sensing coil is directly proportional to each of these variables. Changing any of these variables scales the trends in Figures 9−13 but does not affect the trends. This means that a higher induced voltage can be achieved by increasing any one of these variables. In reality, there are other factors to consider, such as greater resistive losses in a longer wire and system resonance effects. In contrast, the geometry of the primary and sensing coils affects the integrations of (1) and (4) so that there is no linear relationship between the coils' dimensions and the results of the analysis plots. However, the results from the plots can be normalized with respect to the primary coil radius.

To show that the results in Figures 9−13 represent a normalized case that can be applied to scaled coil geometries, the flux through a sensing coil versus misalignment is plotted in Figure 14 for two different coil geometries. In both cases, the sensing coil diameter is 3/4 times the primary coil radius and the height is equal to one primary coil radius. In other words, the sensing coil dimeter and operating height of both plots shown in Figure 14 are the same when normalized with respect to the primary coil radius.

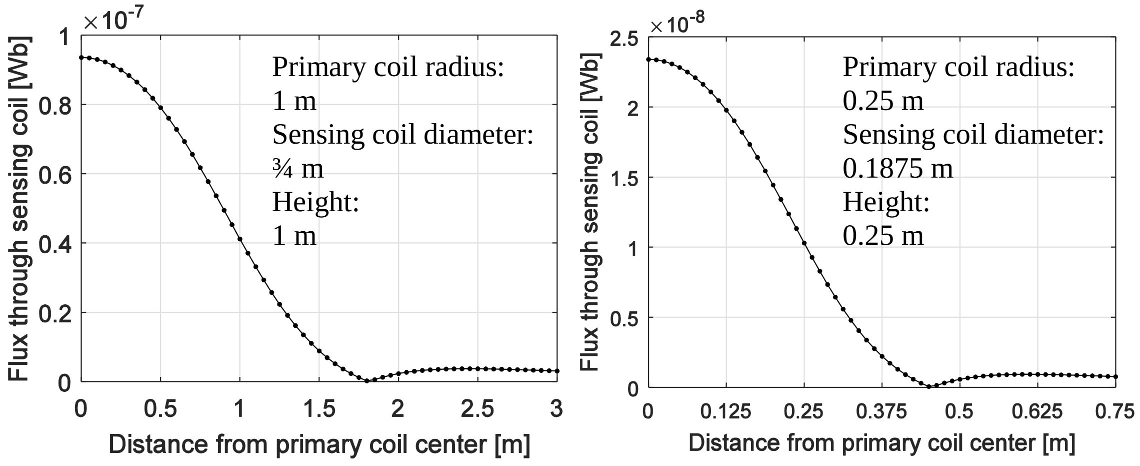


Figure 14. Flux for sensing coils with diameter 3/4 times the primary coil radius, at a height of 1 primary coil radius

The two plots in Figure 14 show identical results, but with scaled axes values. The scaling factor from the normalized to the non-normalized plot is the primary coil radius of the non-normalized case. In short, this scaling is possible because the geometric ratios between the coils are the same in both cases. The importance of this result is that the plots presented in Figures 9−13 can be applied to other systems with scaled geometry, assuming ideal conditions.

The geometric variables in an IPT system that can be scaled are the primary coil radius, sensing coil diameter, and operating height. To find results for any combination of these geometric variables using normalized plots, two non-dimensional numbers can be defined:

- *D:* Ratio of the sensing coil diameter to the primary coil radius
- *H:* Ratio of the operating height to the primary coil radius

The numbers *D* and *H* are normalized quantities with respect to the primary coil radius. Once *D* and *H* are found, they can be used to find a corresponding normalized plot. For example, if the sensing coils are 0.1 m in diameter, the primary coil is 0.4 m in radius, and the working height is 0.3 m, the normalized quantities are D = 0.25, and H = 0.75. Then, the normalized plot for a sensing coil diameter of 0.25 m and height 0.75 m (shown in Figure 15) can be used. Note that the plot in Figure 15 shows small voltage values because the number of sensing coil turns and operating frequency used were the SI unit values. Increasing the frequency and number of coil turns increases the induced voltages.

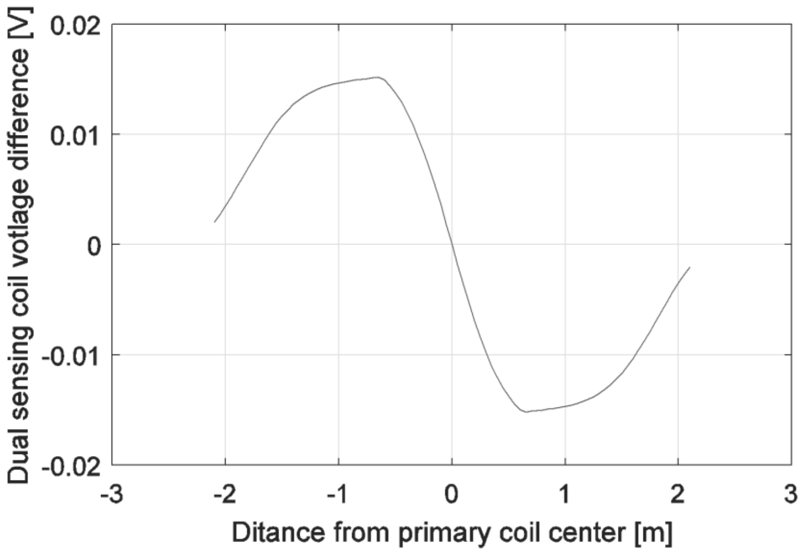


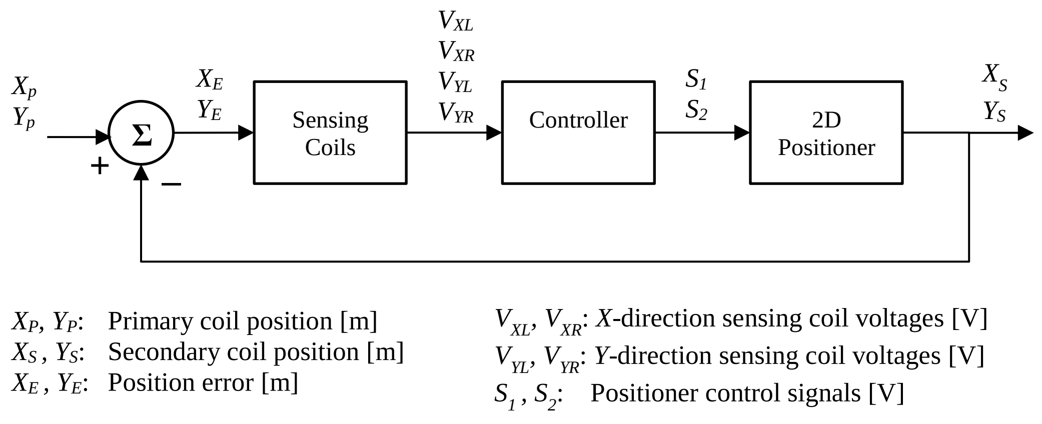
Figure 15. Voltage for dual sensing coils of diameter 0.25 m, primary coil radius 1 m, and height 0.75 m

To find the values for the sensing coil diameter 0.1 m, primary coil radius 0.4 m, and working height of 0.3 m, one only needs to multiply the axes values in Figure 15 by the radius 0.4 m. Thus, the general trends and results provided in the previous analysis can be used to address the behavior of larger (or smaller) IPT systems, neglecting effects of coil resistance and other variables.

The significance of the simulations presented in Figures 9−13 is that the dual sensing coils are expected to function as intended even in larger-scale IPT systems. That is, using the sensing coil configuration of Figure 6 is expected to be a suitable way to sense misalignment direction. The analysis also reveal some limitations, such as a maximum sensing range of about one primary coil diameter and minimum working height of 1/8 primary coil radius. In addition, it is difficult to sense misalignment amount due to the nonlinear voltage trend; a specific voltage maps to two different misalignment amounts. Misalignment amount can be calculated if the sensing coils are used between −1 and 1 primary coil radius only.

2.5 Automatic 2D IPT positioner

Typically, the user of an IPT system is tasked with ensuring good alignment between a primary and secondary coil. If the misalignment can be detected (as has been proposed in this thesis), then an automated mechanism can perform the alignment task. The mechanism should be able to move either the primary or secondary coil to align with the other. Since only planar misalignments are considered in this thesis, the mechanism is only required to move in the two planar directions (*x* and *y* in Figure 5). To reduce misalignment in the *x* and *y* directions, a closed-loop control system is implemented in each misalignment direction (Figure 16). The sensing coils play the crucial role of detecting misalignment and closing the control loop. The voltage subscripts in Figure 16 refer to the sensing coil configuration of Figure 8.



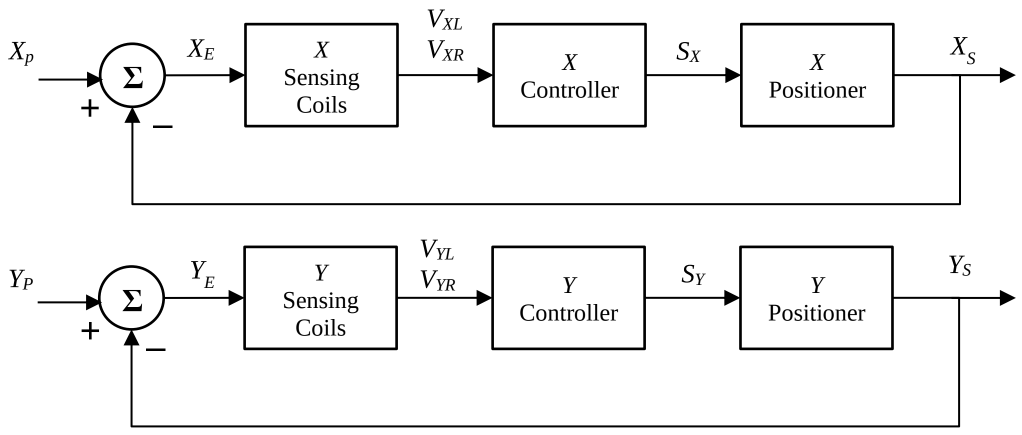
X_p
Y_p
+
–
Σ
X_E
Y_E
Sensing Coils
V_{XL}
V_{XR}
V_{YL}
V_{YR}
Controller
S_1
S_2
2D Positioner
X_S
Y_S
X_P, Y_P: Primary coil position [m]
X_S , Y_S: Secondary coil position [m]
X_E , Y_E: Position error [m]
V_{XL}, V_{XR}: X-direction sensing coil voltages [V]
V_{YL}, V_{YR}: Y-direction sensing coil voltages [V]
S_1 , S_2: Positioner control signals [V]

Figure 16. Closed-loop feedback control scheme for automatic IPT positioning system

In Figure 16, the 2D positioner mechanism can change the *x* and *y* position of the secondary coil, and the primary coil remains stationary. The opposite configuration can also be used. The sensing coils, which remain fixed to the secondary coil, help close the control loop by converting the position errors (X_E and Y_E) into measurable voltages (V_{XL}, V_{XR}, V_{YL}, and V_{YR}). If a proper, stabilizing controller is implemented, this control loop works to drive the position error to zero, at which point the lateral misalignments between the primary and secondary coil are eliminated.

Since there are two degrees of freedom (DOFs) along the plane of lateral misalignments, at least two control signals (S_1 and S_2) are needed to command the positioner. Depending on the positioner dynamics, the control signals can be coupled or decoupled. For instance, rotating robotic arm that operates in polar coordinates would require kinematic transformations that couple the *x* and *y* signals non-linearly. The obvious solution is to avoid control signal coupling and instead use a positioner that can move in the *x* and *y* directions independently.

The signal flow within the sensing coils cannot be decoupled since the *x*-direction sensing coil voltage depends on the *y* position of the coils, and vice-versa. The *x*-*y* dependence of the sensing coils is a result of the 3-dimensional (3D) distribution of the primary coil magnetic field (Figure 7). Nevertheless, the following simplified approach is proposed for use of the sensing coils: ignore cross coupling between the sensing coils and treat the sensing coil output voltages as if they are linear. This approach allows the control system of Figure 16 to be split into *x* and *y* direction flows and allows linear controller design. The decoupled control system is shown in Figure 17.



X_P, Y_P:	Primary coil position [m]	V_{XL}, V_{XR}:	X-direction sensing coil voltages [V]
X_S , Y_S:	Secondary coil position [m]	V_{YL}, V_{YR}:	Y-direction sensing coil voltages [V]
X_E , Y_E:	Position error [m]	S_1 , S_2:	Positioner control signals [V]

Figure 17. Decoupled automatic IPT positioning control system, with *x*- and *y*-direction signal flows

In Figure 17, *x*- and *y*-direction misalignments are treated separately. However, the control system will strive to reduce misalignments in any lateral direction. To demonstrate the expected outcome, consider the misalignment scenarios in Figure 17. The large circle represents the range of significant magnetic flux, which is approximately the same as the primary coil (Figure 9). As long as one of the sensing coils (represented by black circles in Figure 18) is within the large circle, there will be a measurable voltage and misalignment correction can be achieved.

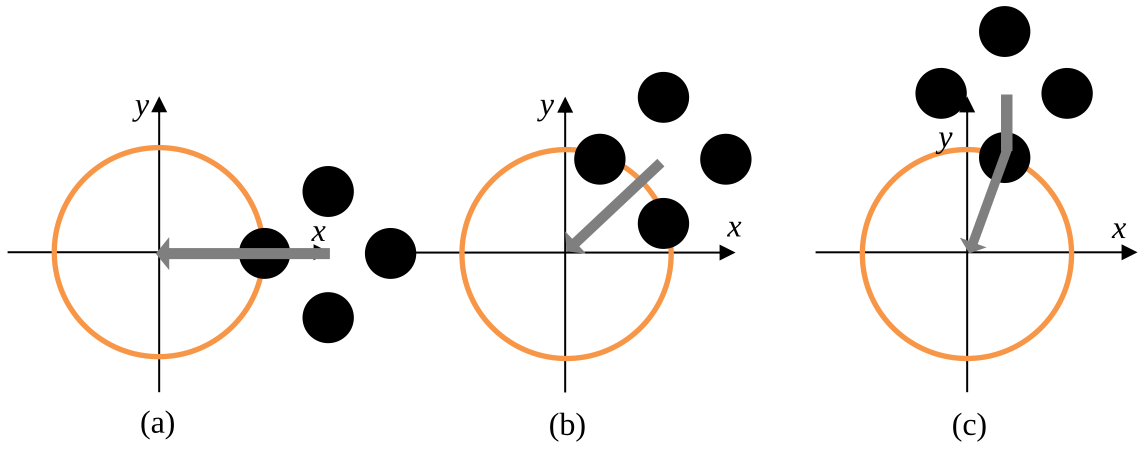


Figure 18. Representative scenarios for sensing coil positions; arrows represent direction of movement by the automatic positioner

Scenario (a) represents the case when there is only misalignment in one direction. According to the results of Section 2.3, misalignments in one direction allow a maximum sensing distance of about one primary coil dimeter. At the edge of the sensing range, only one sensing coil remains within the large circle. A voltage from this single coil will be enough to correct the misalignment and the positioning system movement would follow the arrow shown in the figure.

Scenario (b) represents the case when misalignment occurs equally in the *x* and *y* directions. Here, the sensing coils can only move about 0.7 primary coil diameters before the two closest sensing coils reach the edge of sensing range. Within the range of about 0.7 primary coil diameters, the positioner can sense misalignments in the *x* and *y* directions and would move along the straight arrow shown to restore alignment.

Scenario (c) represents another case where there is misalignment in both directions, but only one sensing coil is within the large circle of magnetic flux. In this case, the positioner would not be able to sense misalignment in both directions initially. However, while correcting misalignment in the *y* direction, other sensing coils will enter the large circle and full alignment can be completed. In this case, the path to alignment is not straight.

From the scenarios in Figure 18 it appears that an *x*-*y* positioning mechanism with the sensing coils and control system of Figure 17 can work as intended to reduce lateral misalignments in an IPT system despite any coupling of the sensing coils. The maximum range of successful misalignment detection is expected to vary between about 0.7 and 1 primary coil diameters depending on the sensing coil orientation with respect to misalignment. The smallest range is expected for the orientation shown in Figure 18 (b), when there is equal misalignment in both the *x* and *y* directions. The 2D misalignment reduction performance of a positioner is tested in Section 6.2, where the *x* and *y* direction control loops are closed independently.

2.6 Mobile robot dynamics and control

To test the application of an IPT coil positioner on a mobile platform, a differential-drive mobile robot platform is developed. This mobile robot design (shown in Figure 19) is used extensively in literature and practice. In differential drive designs, turning is achieved by rotating the left and right wheels of the robot at different speeds. At the front of the robot is a passive ball caster that is low friction and only serves as a contact point to the ground. This simple design allows the mobile robot to move forward, backward, travel in a circle, and even turn in place.

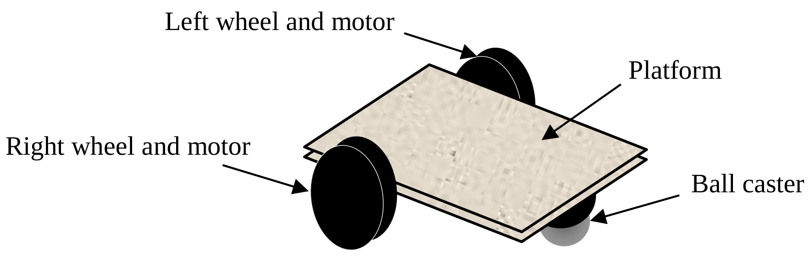


Figure 19. Basic differential drive mobile robot with passive ball caster

The dynamic equations of the mobile robot design pictured in Figure 19 are now presented. Figure 20 defines a global inertial coordinate system (with the basis ***I***, ***J***, ***K***) and the robot's body-fixed frame (with the basis ***i***, ***j***, ***k***). The origin of the body-fixed frame is located at the wheelbase midpoint so that ***k*** is the axis of rotation. The turning of each wheel provides a force to the robot in the ***i*** direction. Forces in the in the ***j*** or ***k*** direction also exist in the general case, but these are not included in Figure 20. The origin of such forces come from contact with the ground.

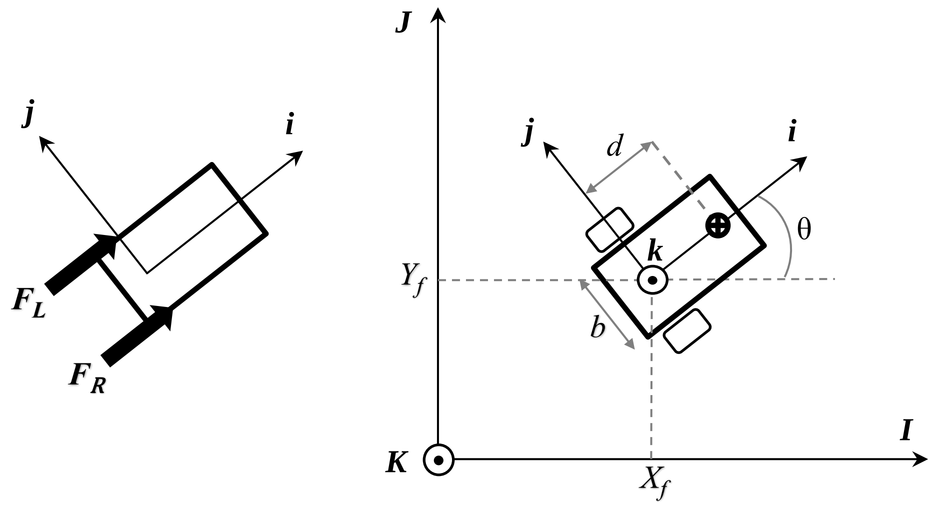


Figure 20. Differential drive mobile robot reference frames and force diagram

Newton's second law of motion for the robot translation gives

$$m\frac{d^2\boldsymbol{R_{cm}}}{dt^2} = (F_L + F_R)\,\boldsymbol{i} \quad , \tag{5}$$

where $\boldsymbol{R_{cm}}$ is the position of the robot center of mass in the global frame and m is the total robot mass. The center of mass position vector can be expressed as

$$\boldsymbol{R_{cm}} = X_f\,\boldsymbol{I} + Y_f\,\boldsymbol{J} + d\,\boldsymbol{i} \tag{6}$$

where X_f and Y_f are the global coordinates of the origin of the robot frame. Then, the time derivatives in the global frame of $\boldsymbol{R_{cm}}$ are

$$\begin{aligned} \frac{d\boldsymbol{R_{cm}}}{dt} &= \frac{dX_f}{dt}\boldsymbol{I} + \frac{dY_f}{dt}\boldsymbol{J} + d\frac{d\theta}{dt}\,\boldsymbol{j} \\ \frac{d^2\boldsymbol{R_{cm}}}{dt^2} &= \frac{d^2X_f}{dt^2}\boldsymbol{\hat{I}} + \frac{d^2Y_f}{dt^2}\boldsymbol{J} + d\frac{d^2\theta}{dt^2}\boldsymbol{j} - d\left(\frac{d\theta}{dt}\right)^2\boldsymbol{i} \quad . \end{aligned} \tag{7}$$

Substituting (7) into (5) gives the translational equation of motion for the mobile robot with force inputs. This translational equation is

$$m\left(\frac{d^2X_f}{dt^2}\boldsymbol{I} + \frac{d^2Y_f}{dt^2}\boldsymbol{J} + d\frac{d^2\theta}{dt^2}\boldsymbol{j} - d\left(\frac{d\theta}{dt}\right)^2\boldsymbol{i}\right) = (F_L+F_R)\,\boldsymbol{i} \;. \tag{8}$$

Finally, coordinate transformations are used to express all terms in the robot body-fixed frame. The equations of motion in the ***i*** and ***j*** directions are

$$\begin{aligned} &m\left(\frac{d^2X_f}{dt^2}\cos\theta + \frac{d^2Y_f}{dt^2}\sin\theta\right) = F_L + F_R + md\left(\frac{d\theta}{dt}\right)^2 \\ \text{and}\quad &m\left(-\frac{d^2X_f}{dt^2}\sin\theta + \frac{d^2Y_f}{dt^2}\cos\theta\right) = -m\,d\frac{d^2\theta}{dt^2}\;, \end{aligned} \tag{9}$$

respectively. The accelerations terms on the left side of (9) correspond to the acceleration of the robot frame origin. These acceleration terms are replaced to give

$$\begin{aligned} m\frac{d^2x_f}{dt^2} &= F_L + F_R + md\left(\frac{d\theta}{dt}\right)^2 \\ m\frac{d^2y_f}{dt^2} &= -m\,d\frac{d^2\theta}{dt^2} \end{aligned} \tag{10}$$

where lower-case variable x_f and y_f denote coordinates of the robot frame origin in the ***i*** and ***j*** directions, respectively.

From Newton's second law for rotating frames, the rotational equation of motion for the mobile robot is

$$\left(I_{cm} + md^2\right)\frac{d^2\theta}{dt^2} = \frac{F_Rb}{2} - \frac{F_Lb}{2} + m\,d\sin\theta\frac{d^2X_f}{dt^2} - md\cos\theta\frac{d^2Y_f}{dt^2}\;. \tag{11}$$

The center of rotation in (11) is the robot ***k*** axis. The last two terms of (11) can be replaced by $md\frac{d^2y_f}{dt^2}$ using the kinematics to give

$$\left(I_{cm} + md^2\right)\frac{d^2\theta}{dt^2} = \frac{F_R b}{2} - \frac{F_L b}{2} - m\,d\frac{d^2y_f}{dt^2} \quad . \tag{12}$$

Equations (10) and (12) define the motion of the mobile robot in its body-fixed frame. Now, these equations are modified so that they agree with expected results. First, note that the acceleration of the origin of the robot frame in the ***j*** direction (expressed as $\frac{d^2y_f}{dt^2}$) should be zero; acceleration in this direction indicates that the robot is slipping in the ***j*** direction. The force that prevents slip in the ***j*** direction is friction between the wheels and the ground, and must be exactly equal to $md\frac{d^2\theta}{dt^2}$ so that $\frac{d^2y_f}{dt^2}$ is zero. Adding this friction force to prevent slip, the equation of motion in the ***j*** direction becomes

$$\frac{d^2y_f}{dt^2} = 0 \quad . \tag{13}$$

Substituting (13) into (12) also gives the updated rotational equation of motion

$$\left(I_{cm} + md^2\right)\frac{d^2\theta}{dt^2} = \frac{F_R b}{2} - \frac{F_L b}{2} \quad . \tag{14}$$

Now, consider the case when F_L is equal and opposite to F_R. With these forces, the robot is expected to turn in place. With such movement, acceleration in the ***i*** direction (expressed as $\frac{d^2x_f}{dt^2}$) should be zero. However, (10) contains the term $md\left(\frac{d\theta}{dt}\right)^2$, which would result in a positive $\frac{d^2x_f}{dt^2}$ even when F_L is equal and opposite to F_R. A positive $\frac{d^2x_f}{dt^2}$ in indicates that the robot is slipping in the ***i*** direction, which is not expected if the robot is turning in place. In reality, friction between the wheels and the ground provide an opposing force which prevents slip in the ***i*** direction in this scenario. The friction force is exactly equal to $md\left(\frac{d\theta}{dt}\right)^2$ to ensure this slipping does not occur. Adding this friction force, the revised equation of motion in the ***i*** direction becomes

$$m\frac{d^2x_f}{dt^2} = F_L + F_R \tag{15}$$

The simplified equations (13) through (15) could have been reached by simply stating that lateral acceleration of the robot is zero and that its angular velocity is small.

Finally, consider the forces F_L and F_R. These forces are provided by wheel motors with their own electromechanical dynamics. However, their response time is relatively fast compared to the mechanical dynamics of the mobile robot platform. To simplify the analysis, assume that the motors provide an instantaneous force proportional to their control signals, which are labeled S_L and S_R. The gain by which these signals are multiplied to get the applied force is labeled K_m. This constant should not be confused with the conventional motor torque constant. Also, to impose a realistic energy decay, damping forces are added to the mobile robot model at the wheel locations. These damping forces represent frictional losses. With damping c_m on the left and right sides of the robot and damping c_r on the ball caster, the final equations of motion for simulation become

$$\begin{aligned} m\frac{d^2x_f}{dt^2} &= K_m S_L + K_m S_R - (2c_m + c_r)\frac{dx_f}{dt} \\ I_f\frac{d^2\theta}{dt^2} &= \frac{S_R K_m b}{2} - \frac{S_L K_m b}{2} - \left(\frac{c_m b^2}{2} + c_r w\right)\frac{d\theta}{dt} \end{aligned} \tag{16}$$

where w is the $\boldsymbol{i}$ coordinate of the ball caster. The moment of inertia expression $I_{cm}+md^2$ is replaced by I_f for conciseness. The $\frac{d^2 y_f}{dt^2}$ equation is omitted since it equals zero. The dynamic equations in (16) serve as a model for the differential-drive mobile robot platform used in experimentation, and the robot parameter values are estimated in Section 6.4.

The mobile robot response given by (16) can be simulated using Matlab Simulink (Figure 21). The inputs to the robot dynamic equations are the motor signals and the outputs are the forward velocity and heading angle of the mobile robot. Note that the forward velocity is in the robot's body-fixed reference frame and the heading angle is in the global frame of reference.

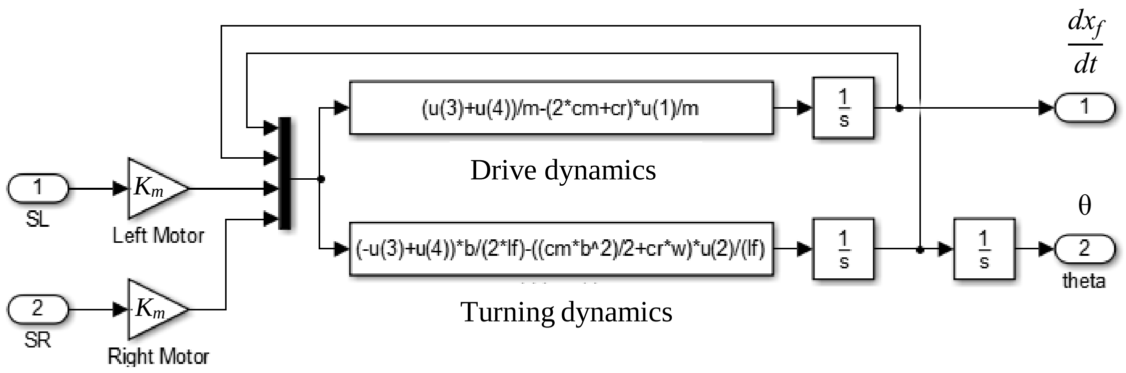


Figure 21. Mobile robot dynamics modeled in Matlab Simulink

Matlab Simulink can also be used to test closed-loop controllers that allow the mobile robot to reach a target location. The simulation setup used for this thesis is shown in Figure 22. The reference inputs are the global coordinates of the target, which remains stationary. The mobile robot begins at the origin of the global coordinate system and initially faces the x direction with a heading angle of 0 rad.

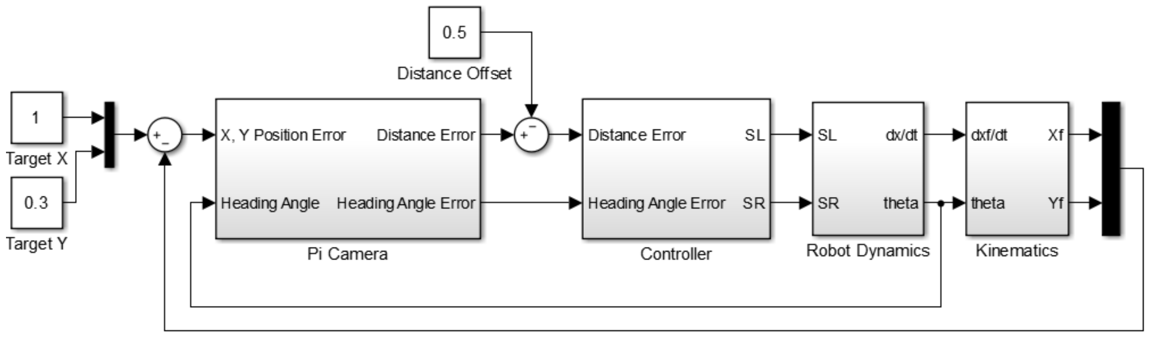
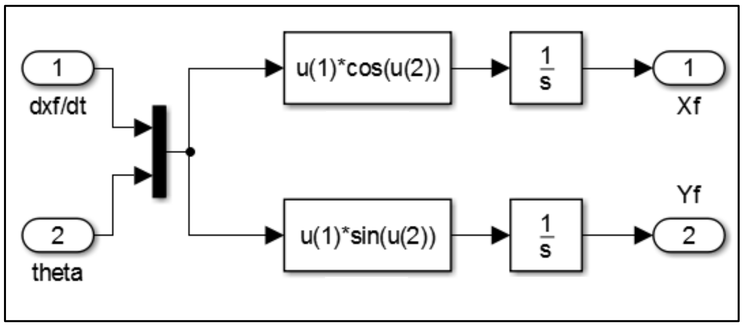


Figure 22. Feedback controller simulation setup using Matlab Simulink

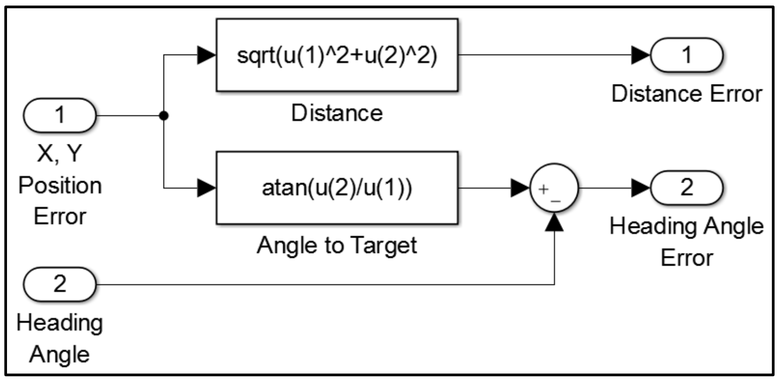
Each of the blocks in Figure 22 are now explained. First, the robot position error must be known. To calculate the position error, the global coordinates of the robot frame (x_f and y_f) are compared with that of the target (Target *X* and *Y*). The global coordinates of the mobile robot is calculated using the kinematic relationships shown in Figure 23.



Kinematics

Figure 23. Kinematic calculations used to find the global mobile robot position

The robot coordinates are fed back and compared to the target coordinates. The control loop is closed by using a camera sensor (the Pi Camera, introduced in Section 3.5) onboard the mobile robot. The Pi Camera outputs a measure of distance and heading angle error to the target as seen by the mobile robot. Ideally, the Pi Camera performs the trigonometric calculations shown in the block of Figure 24. The distance to the target is the Euclidian norm of the global position error vector and the angle to the target is the angle of the positon error vector with respect to the ***I*** direction. Note that the camera sensor has the robot heading angle as an input so that the heading angle error of the robot to the target can be calculated. The heading angle error is the difference between the angle to the target and the actual heading angle of the robot, both with respect to the ***I*** direction.



Pi Camera

Figure 24. Modeled functions of a camera sensor mounted to a mobile robot

The distance and heading angle errors from the camera sensor are used to calculate control signals for the robot. Control design for the mobile robot is presented in Section 5.2. However, it is important to note that there must be coupling between the control signals since the left and right wheels work together to drive and steer the robot. The goal of the controller is to bring the mobile robot to the target—minus an offset distance—so that the mobile robot faces the target directly at the end of its trajectory (see Figure 25).

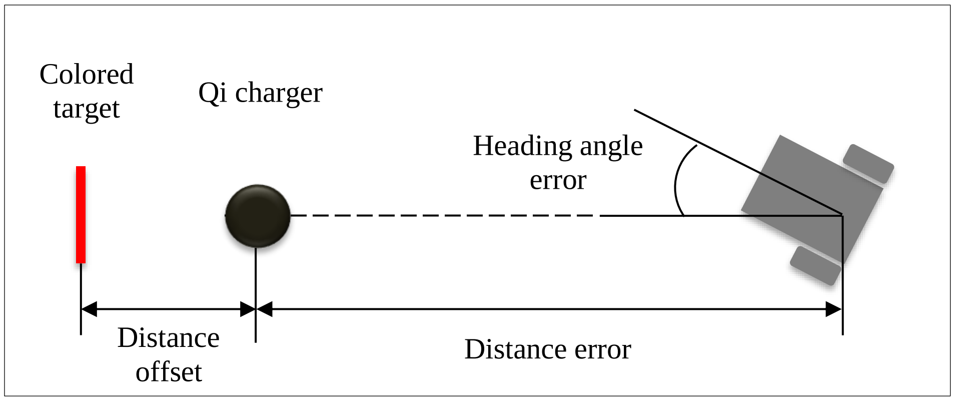


Figure 25. Mobile robot distance and heading angle errors

The mobile robot wireless charging application presented in this thesis is illustrated in Figure 4. According to this scenario, the robot requires the camera sensor to reach the wireless charger location, but the camera sensor is not precise enough to position the robot precisely over the charger. Once the robot arrives close to the wireless charger, the sensing coil measurements can be used to achieve precise alignment. This means that a small steady-state error is allowed with the controller used in Figure 22 as long as the error is small enough that the sensing coils can detect the wireless charger.

CHAPTER III

HARDWARE

Previous analysis gave expected results for the use of dual sensing coils to detect IPT lateral misalignment. The use of such sensing coils in a mobile robot charging application was also proposed. The following hardware is used to verify expected results.

3.1 Experimental IPT System

For experimentation of IPT, a commercial wireless charger is used. The specific model (pictured in Figure 26) is the Yootech T500PB Qi wireless charger. This device contains the power inverter, compensator circuit, and primary coil necessary for an IPT system (Figure 1). The right image of Figure 26 pictures a mobile phone that lies on the Yootech wireless charger. The phone contains the secondary coil, compensator, and power rectifying circuit that supplies a DC voltage to the phone battery charging circuit.

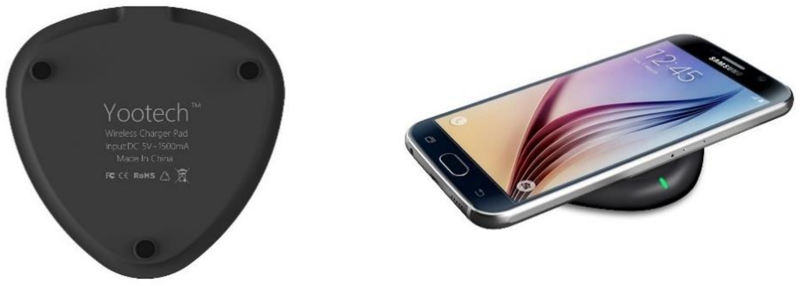


Figure 26. Yootech wireless Qi charger model used for IPT experimentation [33]

A commercial Qi wireless charger is used for two reasons. First, such commercial wireless chargers are a cheap, all-in-one solution for IPT that are easy to power using standard power outlet adapters or batteries. Designing and building a new IPT primary coil circuit is beyond the scope of this thesis and would undoubtedly require an additional analysis. The second reason for using a commercial wireless charger is that the experimental results will be more applicable to existing systems, which likely adhere to the same (or similar) charging standards. The Qi standard is one of the most common wireless charging standards used today.

One significant difference between the commercial charger shown in Figure 26 and the IPT analysis previously presented is the coil construction. In the analysis, a single coil with a fixed diameter was used to simulate the primary coil. However, the primary coil in the Yootech charger is made of ten turns in a flat coil. Due to the wire thickness, this means that the outer loop is at 43 mm diameter and the inner loop is at 20 mm diameter. To address this, the commercial wireless charger can be modeled as a superposition of 10 primary coils with different diameters. If seen as such, the trends from Figures 9−13 remain largely unchanged, with stronger magnetic flux densities near the center of the primary coil.

In lieu of using a mobile phone or similar decive as the secondary coil for the experimental IPT system, a Qi wireless charging receiver is used (Figure 27). This receiver is essentially identical to the circuits found in mobile phones like, thanks to adherence to the same Qi standards. The receiver is directly compatible with the Yootech wireless charger being used. It is necessary to use the receiver in experimentation because the Qi wireless charger is operates only when it senses the presence of a receiver. When the receiver is not present, the Qi wireless charger switches to a pulsing mode.

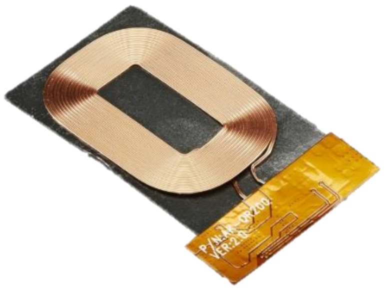


Figure 27. Wireless Qi receiver module [34]

Together, the wireless charger and receiver pictured in Figures 26 and 27 are rated for power transmission up to 5 W. The recommended input voltage to the charger is 5 V at 2 A via a micro-USB port. According to the charger specifications, the maximum power transfer to the end device is 5 W at 5 V and 1 A.

3.2 Sensing coils

Misalignment-sensing coils are constructed for experimentation. The coils are made of enamel-coated 25 AWG copper wire which is wound around a 3D-printed bobbin to create the circular sensing coil loops (Figure 28).

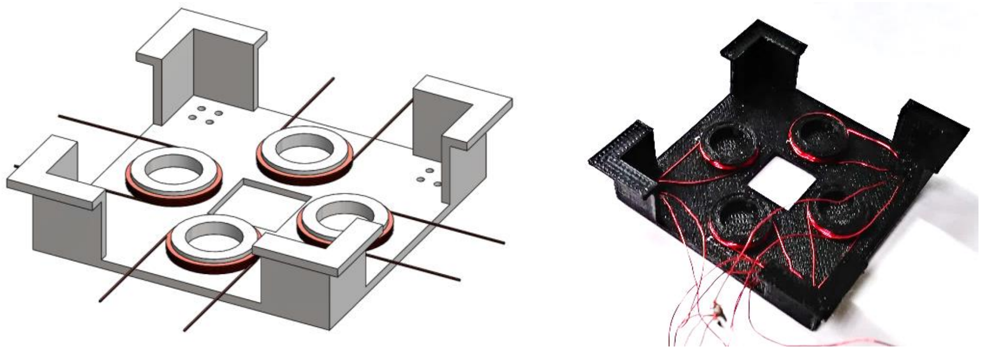


Figure 28. Sensing coils wound on a 3D-printed bobbin; model (left) actual (right)

The bobbin allows four identical sensing coils to be wound in the configuration of Figure 8. Each pair of sensing coils opposite to each other forms one dual sensing coil pair for misalignment detection in one direction. The number of turns used for each sensing coil is 10. This number was decided arbitrarily but also considering the physical consequence of adding more loops (increasing the coil diameter). Using small gage wire allowed for 10 turns to maintain a tight sensing coil diameter.

To measure the voltage magnitude in each a sensing coil, a rectifier circuit is implemented. The rectifier circuit is constructed using a diode rectifier, a smoothing capacitor, and a shunt resistor (Figure 29). To ensure a low voltage drop in the rectification circuit, a Schottky diode (1N5819) is used. The values of the smoothing capacitor and shunt resistor are selected to reduce signal noise but also to allow adequate sampling response time. The values used in the following experiments are 10 μF and 1 kΩ, giving a time constant of 10 ms.

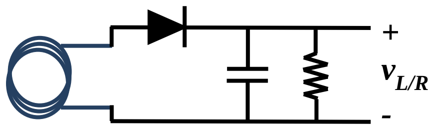


Figure 29. Sensing coil circuit for each sensing coil to measure peak induced voltage

Measurement of the voltage for each sensing coil is done using an Arduino UNO revision 3 microcontroller board (Figure 30). Measurements of the non-rectified voltage and circuit step response is done using a digital oscilloscope (Rigol DS1054Z).

Figure 30. Arduino UNO revision 3 board used for sensing coil voltage measurement and control of actuators [35]

The Arduino UNO board contains 6 input channels dedicated to analog voltage measurement from 0 to 5 V. Each channel features 10-bit analog-to-digital (ADC) conversion, giving a sensing resolution of about 5 mV. One ADC channel is used per sensing coil. The Arduino board also contains 14 digital output pins, 6 of which can be used for pulse-width-modulation (PWM) output. A custom program, written in C, is created for each experiment procedure. The programs are compiled using the Arduino software (version 1.8.1) and are saved in the Arduino board flash memory where it can be executed. In the experiments, some of the Arduino variables are displayed on the computer screen for analysis and plotting. Serial communication with the Arduino is done in real time via the programming USB cable.

In the following experiments, custom Arduino programs are used to measure the voltage in sensing coils, calculate controller signals, and output commands to coil positioning actuators. Complete schematics and Arduino codes are provided for each experiment.

3.3 2D positioner

The first task of this thesis is to demonstrate a way to sense lateral misalignment in an IPT system. The second task is to implement automatic alignment of IPT coils using said misalignment detection capabilities. To complete these tasks, an experimental 2D positioner is designed and constructed (Figure 31). The 2D positioner has the ability to position the sensing coil assembly (Figure 28) along a plane parallel to the primary coil using two stepper motors. The stepper motors are each commanded by an Arduino UNO board, which measures the voltage of the sensing coils and calculates control signals based on the automatic positioning control law (to be presented in Section 5.1).

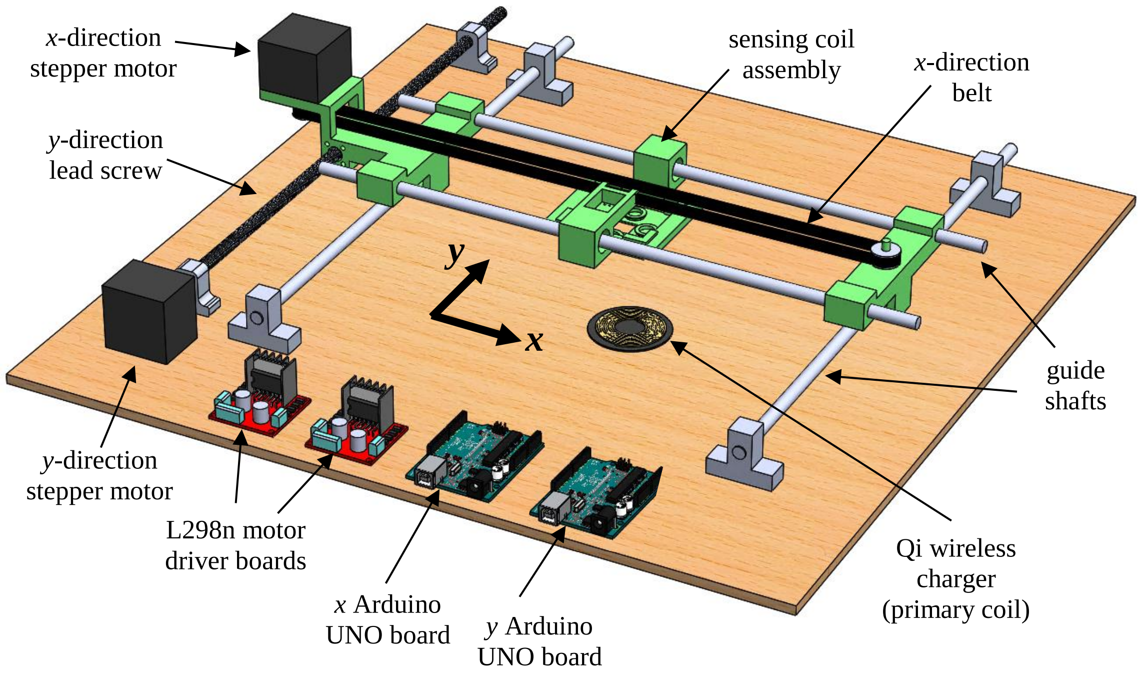


Figure 31. Two-dimensional positioner assembly (wire connections not shown)

The 2D positioner in Figure 31 contains some 3D-printed parts but also makes use of commercially-available components. The leftmost 3D-printed part serves as a mount for the *x*-direction stepper motor, the lead screw nut, and guide shafts. The rightmost 3D-printed part serves as a mount for the guide shafts and the belt pulley. The center 3D-printed part holds the sensing-coil fixture. The belt is also attached to this part to provide *x*-direction movement. As the labeling implies, the *x*-direction stepper motor moves the sensing coil assembly in the *x* direction by turning the belt and the *y*-direction stepper motor moves the sensing coil assembly in the *y* direction by turning the lead screw. Each NEMA 17 stepper motor has a step resolution of 1.8°. The belt pulley has a diameter of 12.7 mm, and the lead screw has 2 mm pitch. This hardware combination gives a positioning resolution of 0.2 mm in the *x* direction and 0.05 mm in the *y* direction if full steps are used with the motors.

The 2D positioner shown in Figure 31 is used for two different experiments. First, the induced voltage in the sensing coils is measured for different misalignment amounts. The results of this experiment are compared to the analysis results from Section 2.3 and can be used to address the first thesis task. In the second experiment, the sensing coils perform as a misalignment sensor to automatically reduce coil misalignment. The sensing coil voltage outputs are measured by the Arduino boards and a controller program provides control signals for the stepper motors. This experiment addresses the second thesis task. The electrical schematics used for the 2D positioner experiments are shown in Figures 32 and 33. The same wiring is used for both the *x* and *y* directions. In addition, there is one wire (not shown in Figures 32 and 33) that goes from the *x*-Arduino digital pin 12 to the *y*-Arduino digital pin 8. The purpose of this wire is explained in the software Section 4.2. The schematics include the rectifier circuits for the sensing coils as well as the wiring of the stepper motors using the H-bridge L298N motor drivers.

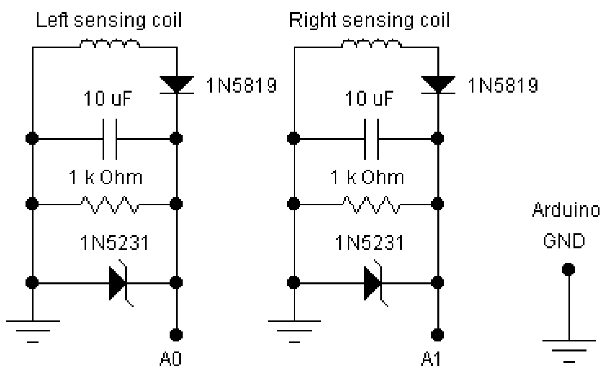


Figure 32. Electrical schematic for either the *x*- or *y*-direction sensing coils in the 2D positioner assembly

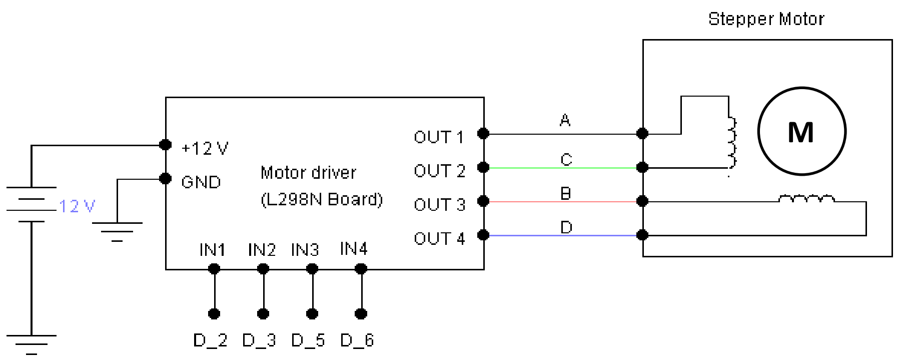


Figure 33. Electrical schematic for 2D positioner stepper motors and L298N H-bridge motor drivers; schematic is the same for the *x*- and *y*-direction motors

The Arduino boards command the stepper motor H-bridge drivers and are powered through the USB port at the logic level of 5 V. Zener diodes are added to the sensing coil circuits (Figure 32) to protect the Arduino board from input voltages higher than 5 V. The 12 VDC supply in Figure 33 is provided by an AC-to-DC wall adapter rated at 1 A output current. The 12 V ground and the Arduino ground are connected.

3.4 Mobile robot

The differential-drive mobile robot platform used for experimentation is pictured in Figure 34. The mobile robot contains the Pi Camera sensor introduced in Figure 22, a sensing coil assembly, and a 1D positioner. The 1D positioner is made with a small dc motor, belt, and pulley. Together, the robot wheels and the mounted 1D positioner provide movement to the secondary coil in the two lateral directions.

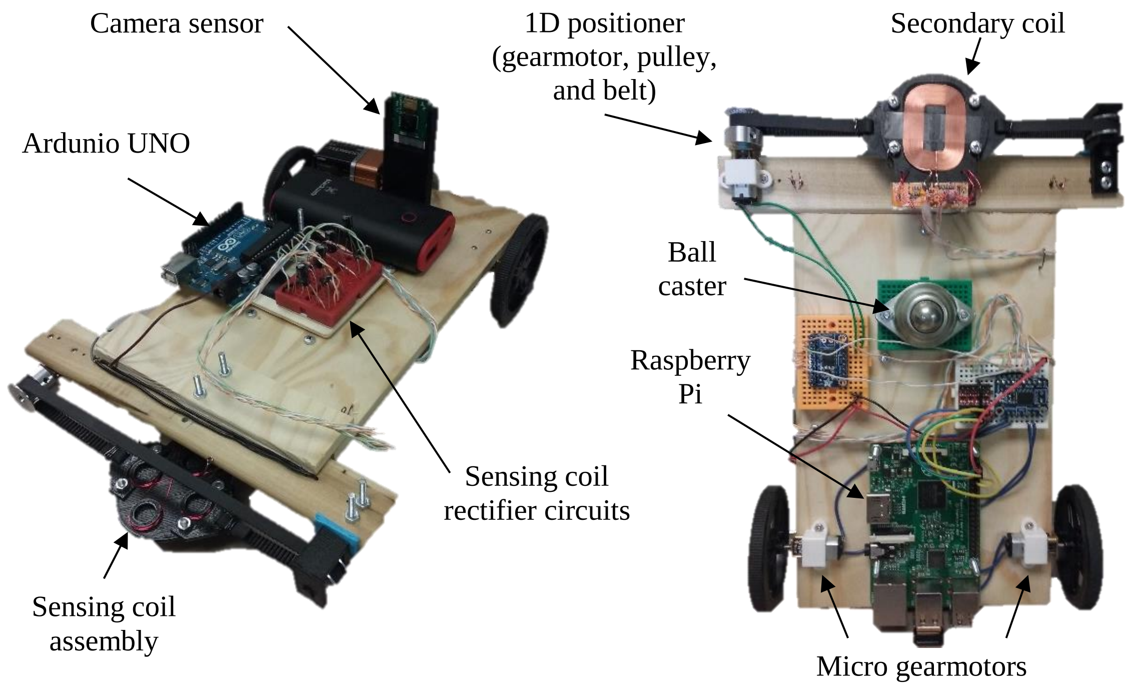


Figure 34. Differential drive platform used for mobile robot application

The mobile robot wheels are turned by small gearmotors with specifications shown in Table 1. The gearmotors are powered by a standard 9 V battery and are controlled using an Adafruit TB6612 H-bridge motor controller.

Table 1. Mobile robot gearmotor specifications

- Generic brand (UPC 701203386617)
- Gear ratio100:1
- No-Load Speed 120 rpm @ 6 V
- Rated Speed 100 rpm @ 6 V
- Rated current 0.21 A @ 6 V
- Rated Torque 0.6 Kg·cm @ 6 V
- D output shaft diameter 3 mm
- Wheels are Pololu 60 mm diameter

The motor board itself is commanded using PWM signals generated by the Arduino board. Each gear motor requires its own PWM signal. Figure 35 shows the motor driver wiring. Another motor driver is added for control of the 1D positioner gear motor.

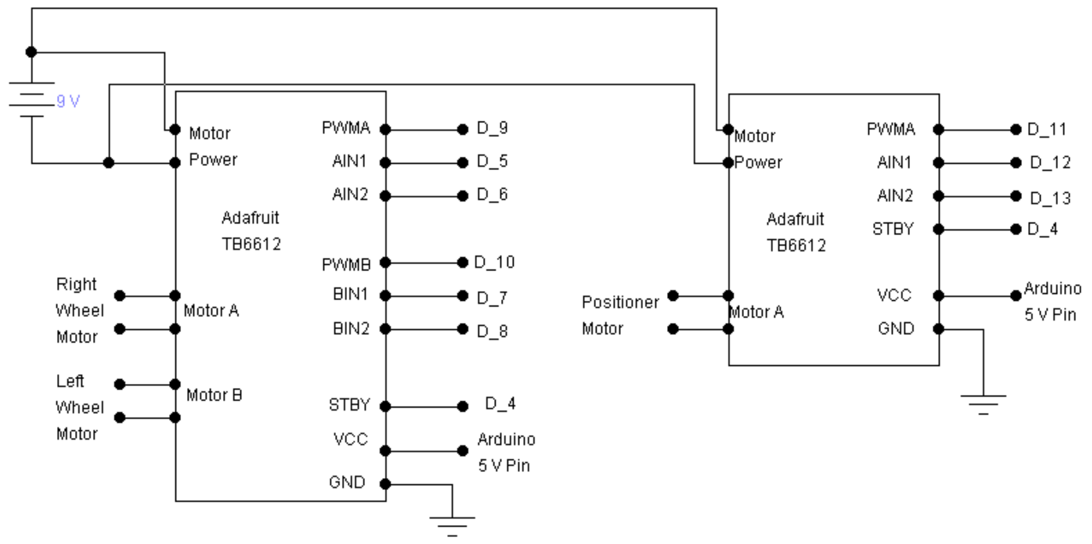


Figure 35. Wiring diagram for PWM control of the differential drive mobile robot

Also mounted on the mobile robot is a set of four sensing coils using the same wiring of Figure 32. These sensing coils are attached to the robot such that they can sense misalignment with respect to a Qi wireless charger on the ground. The sensing coil assembly is attached to the mobile robot using a 1D belt-and-pulley positioning system actuated by a gearmotor.

The mobile robot becomes a 2D IPT coil positioner on its own, performing the same function as the experimental positioner in Section 3.3. According to the scenario presented in Figure 4, after the mobile robot approaches the wireless charging station, it uses the mounted sensing coils to precisely align itself over the Qi wireless charger on the ground. The 1D positioner mounted on the robot facilitates fine positioning by moving the sensing coil assembly in the direction perpendicular to the robot movement (Figure 36). The robot wheels provide the forward and backward positioning motion.

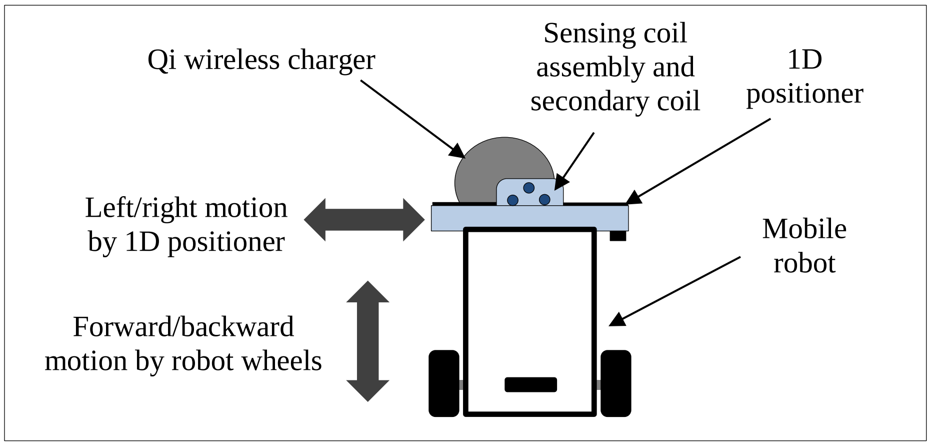

Qi wireless charger
Sensing coil assembly and secondary coil
1D positioner
Left/right motion by 1D positioner
Mobile robot
Forward/backward motion by robot wheels

Figure 36. Mobile robot precise positioning over Qi wireless charger using sensing coils (top view)

3.5 Target-tracking camera

In the mobile robot charging application presented in Figure 4, the mobile robot locates the IPT wireless charger by detecting a colored target that marks the charger location. To detect the target location, a camera sensor with computer-vision software is installed on the robot. The camera used is the Raspberry Pi Camera (revision 1.3), and the images are processed using a Raspberry Pi 2 model B computer (Figure 37). These components are referred to as the Raspberry Pi and Pi Camera, respectively, and they are mounted on the mobile robot as shown in Figure 34. The Pi Camera is placed at the robot wheelbase midpoint and faces forward in the direction of robot movement.

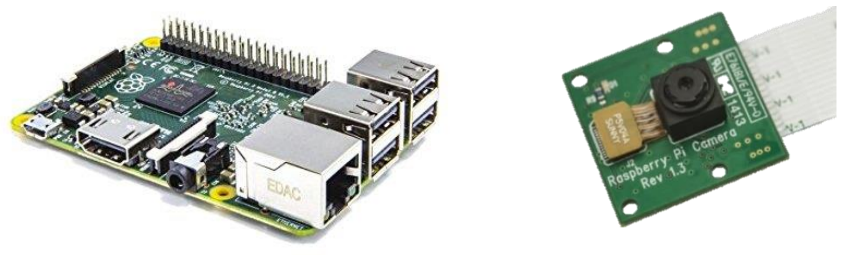


Figure 37. Raspberry Pi 2 B (left) and Raspberry Pi Camera 1.3 (right) [36, 37]

When used with the Raspberry Pi, the Pi Camera can capture images up to 5 megapixels in size and video up to 1080p resolution at 30 frames per second. The image resolution and framerate can be adjusted through software programming. The horizontal field-of-view (FOV) of the camera at full resolution is about 53°.

The Pi Camera closes the loop in the control system of Figure 22 by giving the mobile robot a measure of distance and heading angle error to the target. Thus, the target must remain in the camera FOV for target tracking to occur. The accuracy and sampling rate of the Pi Camera measurements depends on the image-processing script used. The image processing script used for this thesis is presented in Section 4.3.

3.6 Raspberry Pi and Arduino UNO communication

The Raspberry Pi is a small computer that runs on a Linux operating system. There are many ways to program it for use with the Pi Camera. For this thesis, a Python script is used (see Section 4.3 for the script code). The Raspberry Pi, like the Arduino UNO, provides some general-purpose input/output (GPIO) pins. However, the native input/output capabilities of the Raspberry Pi are limited compared to the Arduino. For example, the Raspberry Pi has no ADCs so reading analog voltages is not possible without an external ADC chip. Also, the Raspberry Pi only has two hardware PWM-enabled pins, while the Arduino UNO has 6. Another difference is that the Raspberry Pi operates at the 3.3-V logic level, while the Arduino UNO operates at the 5-V logic level.

The Arduino board’s ability to measure voltages and output several PWM signals makes it ideal for measuring sensing coil voltages commanding electric motors. However, the Arduino board is not capable of processing images captured by the Pi Camera. For this task, the Raspberry Pi is used. To close the control loop of Figure 22, the Raspberry Pi must be able to communicate with the Arduino board, which generates the motor control signals. Therefore, it is necessary to establish a link between the Arduino and Raspberry Pi. Communication between the boards is easily done using the i2C serial communication protocol, which is supported by both the Arduino and Raspberry Pi. A logic-level converter (the Sparkfun converter based on BSS138 n-channel field-effect transistors) is used to convert digital voltage signals between the Arduino and Raspberry Pi to the appropriate voltage levels. The wiring used to establish communication between the Arduino board and the Raspberry Pi using i2C communication is illustrated in Figure 38. The appropriate lines of code used for i2C communication are presented in the programming codes of Sections 4.3 and 4.4.

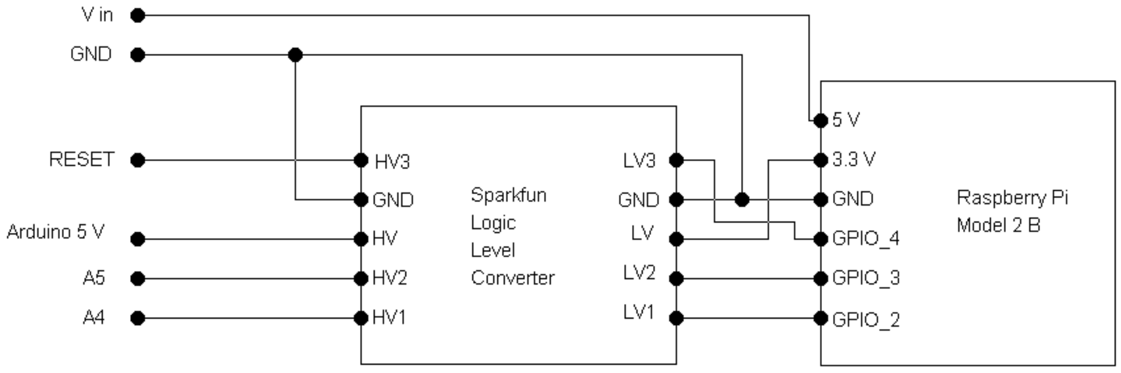


Figure 38. Raspberry Pi and Arduino UNO connections for i2C serial communication using a Sparkfun-brand logic level converter [38]

CHAPTER IV

SOFTWARE

In this chapter, the software and codes that enable the thesis experiments are presented. All the Arduino UNO programs and Raspberry Pi Python scripts presented were written myself specifically for the experiments in this thesis. However, the codes make use of some open-source libraries and functions that are clearly identified.

Omitted in some of the following scripts are print statements which are used to display real-time values of program variables—many times for plotting purposes. For example, the program running time may be printed to time specific functions. The print statements do no alter the functionality of the codes, so it is of no consequence if they are not included in this thesis.

4.1 Sensing-coil voltage measurement

In the first experiment, the voltage of the sensing coils is measured using the Arduino board. The program functions include moving the sensing coils to different *x* and *y* positions and taking voltage measurements at each position. Sensing coil positioning is achieved by controlling the 2D positioner stepper motors (Figure 31).

The program used to accomplish the sensing coil voltage measurements is presented in Table 2. First, a function called *setZero()* gives the user some time to align the IPT system manually. Then, the main function, called *measureVoltages(),* measures the sensing coil voltages at various *x*-direction misalignments. Between each voltage measurement, the *x*-direction stepper motor is moved 5 steps, which gives 1 mm of movement. After the *measureVoltages()* function is complete, the 2D positioner is once again at the aligned position. The program then offsets the sensing coils in the *y* direction by 5 mm and the *measureVoltages()* function runs again. This is repeated for *y*-direction offsets up to 20 mm. The result of this procedure is a set of *x*-direction sensing coil voltage measurements at various *y*-direction offsets. For this experiment, only the *x* Arduino is used and the *y* stepper motor connections are temporarily moved from pins 2,3,5, and 6 on the *y* Arduino to the pins 8,9,10, and 11, respectively, on the *x* Arduino.

Table 2. Arduino program for measuring sensing coil voltages at various *x* and *y* misalignments

```
// Experiment 1: Sensing coil voltage measurement vs. displacement in the x direction at various y
displacements

// Stepper motor setup
#include <Stepper.h>                    // Include standard stepper motor library
Stepper xStepper(200, 2, 3, 5, 6);      // stepperName(steps_per_revolution, in1, in2, in3, in4);
Stepper yStepper(200, 8, 9, 10, 11);

float vxL, vxR, vyL, vyR, xe, ye;       // Voltage reading decimal variables
long t0;

// Required setup function
void setup() {
  Serial.begin(9600);          // Initialize serial communication with the Arduino

  setZero();                   // Initial function for finding zero dislocation location
  xStepper.setSpeed(20);       // Set stepper motor speed in RPM
  yStepper.setSpeed(20);       // Set stepper motor speed in RPM

  measureVoltages();           // Main function for voltage measurements vs x displacements, zero y
displacement

  yStepper.step(100);          // Offset positioner 5mm in y direction
  delay(1000);                 // Small time delay
  measureVoltages();           // Take another set of voltage measurements vs x displacement
```

Table 2 Continued.

```
  yStepper.step(100);          // Offset positioner to 10mm in y direction
  delay(1000);               // Small time delay
  measureVoltages();           // Take another set of voltage measurements vs x displacement

  yStepper.step(100);          // Offset positioner to 15mm in y direction
  delay(1000);               // Small time delay
  measureVoltages();           // Take another set of voltage measurements vs x displacement

  yStepper.step(100);          // Offset positioner to 20mm in y direction
  delay(1000);               // Small time delay
  measureVoltages();           // Take another set of voltage measurements vs x displacement
}

// Required loop function, left empty
void loop() {
}

// This function runs for a few seconds; allows user to center the sensing coil over the primary coil
void setZero()
{
  // Print sensing coil voltages continuously for 15 seconds
  t0 = millis();
  while(millis()-t0<15000){
    // Read voltages for left sensing coils
    vxL = analogRead(A0)*5.0/1023.0;
    vxR = analogRead(A1)*5.0/1023.0;
    vyL = analogRead(A2)*5.0/1023.0;
    vyR = analogRead(A3)*5.0/1023.0;
    // Prints voltage difference for sensing coils
    Serial.print(vxR-vxL, 4);
    Serial.print("   ");
    Serial.println(vyR-vyL, 4);
    delay(100);                  // Small time delay
  }
}

// This is the main function that runs to measure voltages at x misalignments
void measureVoltages()
{
  // Measure negative x displacements
  // Loop for 5 experiment trials
  for(int n=1; n<=5; n++){
    // For each trial, perform following the loop
    for(int i=0; i <= 40; i++){
      vxL = analogRead(A0)*5.0/1023.0;    // Read voltage for left sensing coil
      vxR = analogRead(A1)*5.0/1023.0;    // Read voltage for right sensing coil
      // Print the measured voltages for the current displacement
      Serial.print(-i*5*.0002,4);
      Serial.print("  ");
```

Table 2 Continued.

```
      Serial.print(vxL,4);
      Serial.print("  ");
      Serial.print(vxR,4);
      Serial.print("  ");
      Serial.println(vxR-vxL,4);
      xStepper.step(5);                  // Move 5 steps in negative x direction
      delay(500);                        // Small delay in milliseconds
    }
    xStepper.step(-205);                 // Step backwards to return sensing coils to centered position
    delay(1000);                         // 1 second time delay
  }

  // Measure positive x displacements
  // Loop for 5 experiment trials
  for(int n=1; n<=5; n++){
    // For each trial, perform following the loop
    for(int i=0; i <= 40; i++){
      vxL = analogRead(A0)*5.0/1023.0;    // Read voltage for left sensing coil
      vxR = analogRead(A1)*5.0/1023.0;    // Read voltage for right sensing coil
      // Print the measured voltages for the current displacement
      Serial.print(i*5*.0002,4);
      Serial.print("  ");
      Serial.print(vxL,4);
      Serial.print("  ");
      Serial.print(vxR,4);
      Serial.print("  ");
      Serial.println(vxR-vxL,4);
      xStepper.step(-5);                 // Move a 5 steps in positive x direction
      delay(500);                        // Small delay in milliseconds
    }
    xStepper.step(205);                  // Step backwards to return sensing coils to centered position
    delay(1000);                         // 1 second delay
  }
}
```

4.2 2D positioner driver

The second thesis experiment demonstrates the use of closed-loop control to reduce lateral misalignment in an IPT system. Two Arduino programs are used to do this—one for the *x* and one for the *y* direction. These two programs are nearly identical and are presented in Table 3. At the start of the programs, the sensing coils are given an initial *x*- and *y*-direction misalignment. Then, closed-loop control functions in both

Arduinos run simultaneously to provide 2D alignment. The control law used to align the system, presented in Section 5.2, closes the *x* and *y* loops independently, hence the use of one Arduino for each direction. To ensure that both Arduinos begin operating at the same time, a master-slave relationship is established between them. The wire from digital pin 12 on the *x* Arduino to digital pin 8 on the *y* Arduino is used by the *x* Arduino to trigger the operation of the *y* Arduino. The results of this experiment are time and position data for the *x* and *y* directions of the 2D positioner as the automatic alignment takes place. This data is used to plot the positioner trajectory as alignment takes place.

Table 3. Arduino programs for closed loop control of the 2D positioner to eliminate IPT lateral misalignment

```
// Experiment 2: Closed-loop control of two-dimensional positioner to eliminate IPT misalignment in x
and y directions
// X direction sensing and driver, MASTER to Y direction Arduino

// Stepper motor setup
#include <Stepper.h>                          // Include standard stepper motor library
Stepper xStepper(200, 2, 3, 5, 6);              // stepperName(steps_per_revolution, in1, in2, in3, in4);

// Initialize variables
int xStep, xSpeed, timeDelay;                   // Stepper motor steps and time delay
float vxL, vxR, xe, cmX, xPos;                  // Voltage and position variables
float vthresh = 0.05;                         // Threshold voltage for controller dead band
long t0;                                    // Time storage variable (to set time zero)

// Required setup function
void setup() {
  Serial.begin(250000);                        // Initialize serial communication with the Arduino
  pinMode(12, OUTPUT);                          // Pin to command slave Y Arduino

  setZero();                                // Function allows user to center IPT coils
  xStepper.setSpeed(200);                      // Set max stepper motor speed in RPM

  // Set intial misalignmen amounts in cm
  cmX = -2;
```

Table 3 Continued.

```
// Step motor creates initial misalignment
  xStepper.step(cmX*50);

  // Keep track of position
  xPos = cmX/100;

  // Wait for keyboard input to start program, send command to Y Arduino when start
  Serial.println("Waiting for go command");
  while(Serial.available()<1){}
  digitalWrite(12, HIGH);
  t0 = millis();                        // Set zero time
}

// Loop function, runs indefinitely
void loop(){
  alignX();
}

// This function runs for a few seconds; allows user to center the sensing coil over the primary coil
void setZero()
{
  // Print sensing coil voltages continously for 15 seconds
  t0 = millis();
  while(millis()-t0<15000){
    // Read voltages for left sensing coils
    vxL = analogRead(A0)*5.0/1023.0;
    vxR = analogRead(A1)*5.0/1023.0;
    // Prints voltage difference for sensing coils
    Serial.println(vxR-vxL, 4);
    delay(100);               // Small time delay
  }
}

// X direction automatic aligning function
void alignX(){
  // Print current time and x position
  Serial.print((millis()-t0)/1000.0, 4);
  Serial.print("  ");
  Serial.println(xPos, 4);

  // Read voltages from sensing coils
  vxL = analogRead(A0)*5.0/1023.0;
  vxR = analogRead(A1)*5.0/1023.0;

  // Caclulate voltage difference
  xe = vxR-vxL;

  // Set time delay in milliseconds
  timeDelay = max((0.75-abs(xe))*20, 0);
```

Table 3 Continued.

```
  delay(timeDelay);

  // Set step direction for x stepper motor based on error sign
  if(xe>vthresh){
    xStep = -1;
  }
  else if(xe<-vthresh){
    xStep = 1;
  }
  else{
    xStep = 0;
  }

  // Send step to stepper motor
  xStepper.step(xStep);

  // Keep track of x position
  xPos = xPos+xStep*0.0002;
}
```

```
// Y direction positioner driver, SLAVE to X direction Arduino

// Stepper motor setup
#include <Stepper.h>                    // Include standard stepper motor library
Stepper yStepper(200, 2, 3, 5, 6);          // stepperName(steps_per_revolution, in1, in2, in3, in4);

// Initialize variables
int yStep, ySpeed, timeDelay;             // Stepper motor steps and time delay
float vyL, vyR, ye, cmY, yPos;             // Voltage and position variables
float vthresh = 0.05;                  // Threshold voltage for controller dead band
long t0;                          // Time storage variable (to set time zero)

// Required setup function
void setup() {
  Serial.begin(250000);                 // Initialize serial communication with the Arduino
  pinMode(8, INPUT_PULLUP);

  setZero();                        // Function allows user to center IPT coils
  yStepper.setSpeed(200);               // Set max stepper motor speed in RPM

  // Set intial misalignment amount in cm
  cmY = 2;

  // Step motor creates initial misalignment
  yStepper.step(cmY*200);
```

Table 3 Continued.

```
  // Keep track of position
  yPos = cmY/100;

  // Wait for command from X driver to start at the same time
  while(digitalRead(8)==LOW){}
  t0 = millis();                              // Set zero time
}

// Loop function, runs until X driver is reset
void loop(){
  while(digitalRead(8)==HIGH){
    alignY();
  }
}

// This function runs for a few seconds; allows user to center the sensing coil over the primary coil
void setZero()
{
  // Print sensing coil voltages continously for 15 seconds
  t0 = millis();
  while(millis()-t0<15000){
    // Read voltages for left sensing coils
    vyL = analogRead(A0)*5.0/1023.0;
    vyR = analogRead(A1)*5.0/1023.0;
    // Prints voltage difference for sensing coils
    Serial.println(vyR-vyL, 4);
    delay(100);
  }
}

// Y direction automatic aligning function
void alignY(){
  // Print current time and y position
  Serial.print((millis()-t0)/1000.0, 4);
  Serial.print("  ");
  Serial.println(yPos, 4);

  // Read voltages from sensing coils
  vyL = analogRead(A0)*5.0/1023.0;
  vyR = analogRead(A1)*5.0/1023.0;

  // Caclulate voltage difference
  ye = vyR-vyL;

  // Set time delay in milliseconds
  timeDelay = max((0.75-abs(ye))*20, 0);
  delay(timeDelay);

  // Set step direction for y stepper motor based on error sign
```

Table 3 Continued.

```
  if(ye>vthresh){
    yStep = 4;
  }
  else if(ye<-vthresh){
    yStep = -4;
  }
  else{
    yStep = 0;
  }

  // Send step to stepper motor
  yStepper.step(yStep);

  // Keep track of y position
  yPos = yPos+yStep*0.00005;
}
```

4.3 Target-tracking camera

To give the mobile robot autonomy to reach the Qi wireless charger, a Raspberry Pi Camera (Figure 37) and basic computer-vision software is implemented via a Python script on the Raspberry Pi. To aid in this task, the open-source computer-vision software, called OpenCV, is installed. This software includes standard libraries for image manipulation and analysis.

For completeness, it is noted that installing the operating software on the Raspberry Pi is a lengthy process that is beyond the scope of this thesis. Suffice it to say that, in this thesis, the operating system on the Raspberry Pi is Raspbian Wheezy with OpenCV 3 and scripts that execute in Python 3. Furthermore, the Raspberry Pi is programmed using the text-based user terminal using standard commands. As an example, the command *sudo nano my_script.py* is used to edit a python script with the name 'my_script.' Accessing the Raspberry Pi command line can be done via an Ethernet cable, WiFi, or by attaching a monitor, keyboard, and mouse directly into the Raspberry Pi. The various methods for installing the software or programming the Raspberry Pi are not detailed in this thesis, and there are many free resources available on the Internet [39‒41].

An image processing script that allows the Pi Camera to detect the colored target and calculate the distance and heading angle error to the target is presented in Table 4. Since the Raspberry Pi must communicate with the Arduino to control the mobile robot (see Section 3.6), this script includes proper i2C communication commands.

Table 4. Raspberry Pi Python script for target-tracking with distance and angle calculations

```
# Target-tracking script with distance and angle calculations

# Import necessary packages
from picamera import PiCamera                 # Camera
from picamera.array import PiRGBArray        # Camera RGB array
import time                                   # Time functions (such as time delay)
import cv2                                    # Computer vision software
import numpy                                  # Allow manipulation of numpy arrays (matrices)
import smbus                                  # Serial communication bus for i2c
import math                                   # Enable math functions

# Setup Arduino i2c communication
bus = smbus.SMBus(1)

# Initialize variables calculated by camera
distance = 0
angle = 0

# Create hsv threshold values for pink post-it target
hmin = 150
hmax = 200
smin = 130
smax = 230
vmin = 100
vmax = 255

# Initialize the camera
camera = PiCamera()
```

Table 4 Continued.

```
width = 320
height = 120
camera.resolution = (width,height)
camera.iso = 400
time.sleep(2)
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = 'off'
g = camera.awb_gains
camera.awb_mode = 'off'
camera.awb_gains = g
rawCapture = PiRGBArray(camera, size = (width,height))

# Target-finding function using opencv image processing
def findTarget(image):
        hsv_image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
        # Filter image using hsv threshold values defined previously
        h,s,v = cv2.split(hsv_image)
        hf = cv2.inRange(h,numpy.array(hmin),numpy.array(hmax))
        sf = cv2.inRange(s,numpy.array(smin),numpy.array(smax))
        vf = cv2.inRange(v,numpy.array(vmin),numpy.array(vmax))
        filt_image = cv2.bitwise_and(hf, cv2.bitwise_and(sf,vf))

        #cv2.imshow("Filtered",filt_image)
        #key = cv2.waitKey(1) & 0xFF

        # Find contours in filtered image
        cont_image,cont,hierarchy = cv2.findContours(filt_image, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        cont = sorted(cont, key = cv2.contourArea, reverse = True)
        if len(cont)>0:
                for c in cont:
                        per = cv2.arcLength(c, True)    # Estimate perimeter of contour
                        approx = cv2.approxPolyDP(c, 0.075*per, True)   # Create approximate
polygon that best fits contour
                        # Check number of sides to polygon. If four, do the following:
                        if len(approx) == 4:
                                area = cv2.contourArea(c)       # Calcualte area of contour
                                mom = cv2.moments(approx)       # Calcualte the moments of the
polygon
                                if mom['m00'] != 0:
                                # Calculate center of polygon, x and y position
                                        x = int(mom['m10']/mom['m00'])
                                        y = int(mom['m01']/mom['m00'])
                                        return[True, x, y, area, approx]
                                break
        return[False, 0, 0, 0, 0]

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        image = frame.array                                             # Image frame array in bgr
```

Table 4 Continued.

```
        [foundTarget,x,y,area,approx] = findTarget(image)    # Send frame to target track function
        rawCapture.truncate(0)                                  # Truncate capture for next
frame
        if foundTarget:
                #cv2.drawContours(image, [approx], -1, (0, 255, 0))  # Draw target polygon
                #cv2.circle(image, (x,y), 2, (255,255,255), -1)           # Draw target center

                # Calculate distance and angle to target
                distance = math.sqrt(700/area)
                angle = 53/2-53*x/width

                # Send data to Arduino. 2 bytes sent. Each byte from -127 to 127
                # Convert values to integers
                distance = int(distance*65)
                angle = int(angle)

                # Send data to Arduino
                try:
                        bus.write_i2c_block_data(0x08, distance, [angle])
                # If error sending data, Arduino may have frozen. Reset Arduino by making GPIO 4
low.
                except IOError:
                        print("Error sending data. Flushing i2c buffer...")

        # Display image
        #cv2.imshow("Image",image)
        #key = cv2.waitKey(1) & 0xFF

        #if key == ord("q"):
        #       break
```

The Python script presented for image tracking presented in Table 4 consists of the following three main parts:

1. Image processing to identify the target in the image frame
2. Estimation of distance and direction of the target as seen by the robot
3. Data transmission of the distance and angle data to the Arduino board

These three parts are executed in order every time a new image frame is captured. The first part is an image processing algorithm that identifies the colored target in the image frame. This part calculates the colored target pixel area and pixel location in the image. The specific steps used to accomplish the image processing are presented in Table 5. These steps are implemented in the function *findTarget()* in the script of Table 4.

Table 5. Image processing steps for colored target-tracking

1.	Capture an image as a 3D matrix using RGB values. For example, with a resolution of 320×240 pixels, the image is saved in three matrices each 320 elements wide and 240 elements tall. One matrix stores the red intensity values (from 0 to 255), the other stores green, and the last stores the blue.
2.	Convert the image from RGB to hue/saturation/value (HSV) format. This format has the same structure as the RGB image, but makes it easier to identify different objects in the image. Converting to HSV is not necessary but yields more consistent image tracking in some cases.
3.	Filter the HSV image according to pre-determined threshold numbers for hue, saturation, and value. These numbers are selected so that the filtered image contains only the colored target of interest.
4.	Create a contour for each area in the image which survived filtering.
5.	Sort through the contours and identify which the colored target is. This is done by fitting a polygon to the contour in question and checking if the polygon has four vertices (the target is rectangular).
6.	The area and location of the contour in the captured image is approximated using pixel area moment calculations.

To find the HSV values for the step 3 in Table 5, a short script is executed on the Raspberry Pi. This script, which is presented Table 6, that allows the HSV values to be adjusted in real time to show the effects on the filtered image.

Table 6. Raspberry Pi Python script for experimentation with HSV filtering values

```
# Import packages for image processing
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2
import time
import numpy

# Initialize the Pi camera
width = 320
height = 240
camera = PiCamera()
camera.resolution = (width, height)
rawCapture = PiRGBArray(camera, size=(width, height))

# Create trackbars that user can edit
def nothing(x):
    pass
cv2.namedWindow('Tracking')
cv2.createTrackbar('hmin','Tracking',150,255,nothing)
cv2.createTrackbar('hmax','Tracking',200,255,nothing)
cv2.createTrackbar('smin','Tracking',130,255,nothing)
cv2.createTrackbar('smax','Tracking',230,255,nothing)
cv2.createTrackbar('vmin','Tracking',100,255,nothing)
cv2.createTrackbar('vmax','Tracking',255,255,nothing)

# Capture image frames in bgr format
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        # Create an array representing the image
        image = frame.array
        # Display original image
        cv2.imshow("Original", image)
        key = cv2.waitKey(1) & 0xFF

        # Convert the image to HSV
        image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV)

        # Update HSV threshold values from track bars
        hmin = cv2.getTrackbarPos('hmin','Tracking')
        hmax = cv2.getTrackbarPos('hmax','Tracking')
        smin = cv2.getTrackbarPos('smin','Tracking')
        smax = cv2.getTrackbarPos('smax','Tracking')
        vmin = cv2.getTrackbarPos('vmin','Tracking')
        vmax = cv2.getTrackbarPos('vmax','Tracking')

        # Filter using HSV threshold values
        h,s,v = cv2.split(image)
        hf = cv2.inRange(h,numpy.array(hmin),numpy.array(hmax))
        sf = cv2.inRange(s,numpy.array(smin),numpy.array(smax))
        vf = cv2.inRange(v,numpy.array(vmin),numpy.array(vmax))
```

Table 6 Continued.

```
filt_image = cv2.bitwise_and(hf, cv2.bitwise_and(sf,vf))

# Display HSV and filtered image
cv2.imshow("HSV", image)
key = cv2.waitKey(1) & 0xFF
cv2.imshow("Filtered", filt_image)
key = cv2.waitKey(1) & 0xFF

# Clear the capture to allow next frame capture
rawCapture.truncate(0)
```

The second part of the Python script presented in Table 4 estimates the distance and heading angle to the target as seen by the mobile robot. The estimates are based on the target pixel area and location in the image frame, respectively. A larger target pixel area indicates that the robot is closer to the target. Similarly, the horizontal position of target center indicates if the robot is facing the target. The respective calculations performed in this part of the target-tracking script are discussed in Section 6.3.

The last part of the Python script in Table 4 is responsible for sending the calculated distance and angle to the Arduino via i2C serial communication. Two signed bytes of data are sent consecutively. The first byte is labeled *distance*, and its value is an integer that ranges from −127 to 127. The second byte is labeled *angle*, and it also ranges from −127 to 127. Note that this integer range makes sense for the unit of angles (the field of view of the Pi Camera is about 53 degrees), but it does not make sense for distance, which may range up to a few meters at most. For this reason, the calculated distance is scaled by a factor of 65 in the Raspberry Pi script (Table 4) to better make use of the −127 to 127 integer range. The distance value received by the Arduino is divided by 65 in the

Arduino program (Table 7) to convert back to the original unit of meters. The scaling factor of 65 was selected so that a 2 m distance is scaled to near 127.

Occasionally, serial transmission to the Arduino might fail for unknown reasons. Normally, this failure aborts the target-tracking program. Rather than exit the program, a try-catch statement is used to detect this occurrence. The try-catch statement gives some time for the Arduino to flush the serial data buffer and reset serial communication. This reset takes only a moment to complete and the script continues to run.

4.4 Mobile robot driver

The Arduino board is responsible for receiving bytes of data from the Raspberry Pi and commanding the mobile robot according to a control law—also programmed in the Arduino. The Arduino board is also responsible for reading the sensing coil voltages and commanding the 1D positioner motor (see Figure 34). These tasks are accomplished in the Arduino program presented in Table 7.

Table 7. Arduino program for driving the mobile robot and 1D positioner

```
//Experiment 3: Automatic positioning of mobile robot over Qi wireless charger using Raspberry Pi
camera and sensing coils

#include <Wire.h>                    // Include library used for i2C serial communication

// Define pins for robot motor control, based on Adafruit TB6612 H-bridge driver board
#define stdby 4
#define rightMotorIn1 5
#define rightMotorIn2 6
#define rightMotorPWM 9
#define leftMotorIn1 7
#define leftMotorIn2 8
#define leftMotorPWM 10

// Define pins for positioner motor control
#define positionerIn1 12
```

Table 7 Continued.

```
#define positionerIn2 13
#define positionerPWM 11

float vxL, vxR, vyL, vyR, xe, ye; // Sense coil voltage reading variables

// Initialize variables for Rapberry Pi communication
// Char are signed 8bit values sent by Raspberry Pi master (-127 to 127)
char distance = 0;                    // Distance to target from Raspberry Pi camera
char angle = 0;                       // Angle to target from Raspberry Pi camera
unsigned long currentTime, lastTime;    // Time variables for keeping track of communication times

// Controller variables used for raching target
float dist_setpoint = 0.5;             // Distance setpoint [m]
float p_distance = 250;                // Proportional gain for distance [pwm/m]
float p_angle = 10;                    // Proportional gain for angle [pwm/degree]
float dist_error = 0;                  // Distance error variable
int minPWM = 40;                       // Set minimum PWM value for motors
int maxPWM = 150;                      // Set maximum PWM value for motors
float dist_thresh = 0.01;              // Set distance threshold for control deadband
int angle_thresh = 0;                  // Set angle threshold for control deadband
int printing = 1;                      // Variable to switch between target reach and IPT position printing

void setup() {
  Wire.begin(8);                       // Join i2c bus with address #8
  Wire.onReceive(receiveEvent);        // Function to run when receive data from Raspberry Pi

  // Setup pin umbers for positioner motor driver
  pinMode(stdby, OUTPUT);
  pinMode(rightMotorIn1, OUTPUT);
  pinMode(rightMotorIn2, OUTPUT);
  pinMode(leftMotorIn1, OUTPUT);
  pinMode(leftMotorIn2, OUTPUT);
  pinMode(rightMotorPWM, OUTPUT);
  pinMode(leftMotorPWM, OUTPUT);

  // Enable motor drivers
  digitalWrite(stdby, HIGH);
  Serial.begin(9600);
}

// Main loop function, runs indefinitely
void loop() {
  // If statement to determine if target has been reached
  if(abs(dist_error) > dist_thresh || abs(angle) > angle_thresh){
    // Lines to run at beginning of reach target routine
    if(printing == 1){
      Serial.println("Reach Target [Time  Dist_error  Angle_error]");
      // Set error thresholds and printing variable until target is reached
      dist_thresh = 0.01;
```

Table 7 Continued.

```
      angle_thresh = 0;
      printing = 0;
    }
    reachTarget();                    // Function for mobile robot to reach Qi target
  }
  // Target has been reached, enable positioning
  else{
    // Lines to run at beginning of positioning routine
    if(printing == 0){
      Serial.println("Positioning [Time  xe_voltage  ye_voltage]");
      // Set error thresholds and printing variable until reach target is needed again
      dist_thresh = 0.05;
      angle_thresh = 7;
      printing = 1;
    }
    positioning();
  }

  // Stop everything if no data is received from Raspberry Pi for 1 sec
  currentTime = millis();              // Get current time
  if((currentTime-lastTime)>1000){
    dist_error = 0;
    angle = 0;
  }
}

// Receive 2 bytes from Raspberry Pi, each signed from -127 to 127
void receiveEvent(int n) {
  // If else statement used to catch problems with data transmission
  // If more or less than 2 bytes, flush data in else statement
  if(n==2){
    lastTime = millis();
    distance = Wire.read();
    angle = Wire.read();
    dist_error = distance/65.0-dist_setpoint;
    // Print distance and angle data if target reach routine is running
    if(printing == 0){
      Serial.print(millis()/1000.0, 4);
      Serial.print("  ");
      Serial.print(dist_error, 4);
      Serial.print("  ");
      Serial.println((int)angle);
    }
  }
  else{
    while(Wire.available()>0){
      char error = Wire.read();
    }
  }
}
```

Table 7 Continued.

```
// Function for mobile robot reaching target using Raspberry Pi camera data
void reachTarget() {
  // Drive control laws based on distance and angle error
  int leftPWM = dist_error*p_distance - angle*p_angle;
  int rightPWM = dist_error*p_distance + angle*p_angle;
  motor(leftPWM, leftMotorIn1, leftMotorIn2, leftMotorPWM);
  motor(rightPWM, rightMotorIn1, rightMotorIn2, rightMotorPWM);
  delay(10);
}

// Function for precise positioning of mobile robot above Qi wireless charger
void positioning(){
    // Measure voltage of sense coils
    vxL = analogRead(A0)*(5.0/1023.0);
    vxR = analogRead(A1)*(5.0/1023.0);
    vyL = analogRead(A2)*(5.0/1023.0);
    vyR = analogRead(A3)*(5.0/1023.0);

    // Calculate voltage differences
    xe = vxR-vxL;
    ye = vyR-vyL;

    // Print voltage differences
    Serial.print(millis()/1000.0, 4);
    Serial.print("  ");
    Serial.print(xe, 4);
    Serial.print("  ");
    Serial.println(ye, 4);

    // Set deadband to avoid limit cycling
    if(abs(xe)<0.1){
      xe = 0;
    }
    if(abs(ye)<0.1){
      ye = 0;
    }

    // Drive controller for y-direction misalignment
    int drivePWM = ye*60;
    motor(drivePWM, leftMotorIn1, leftMotorIn2, leftMotorPWM);
    motor(drivePWM, rightMotorIn1, rightMotorIn2, rightMotorPWM);
    // Positioner controller for x-direction misalignment
    int pos = xe*100;
    motor(pos, positionerIn1, positionerIn2, positionerPWM);
    delay(10);
}

// Motor drive helper function, for use with Adafruit TB6612 Motor Driver Board
// sp = -255 to 255, sign indicates direction of movement
```

Table 7 Continued.

```
void motor(int sp, int in1, int in2, int pin) {
    // Set driver to stop motor of speed is zero
    if(sp==0){
        digitalWrite(in1,LOW);
        digitalWrite(in2,LOW);
    }
    // Set driver to positive motor direction
    else if(sp>0){
        digitalWrite(in1,LOW);
        digitalWrite(in2,HIGH);
    }
    // Set driver to negative motor direction
    else if(sp<0){
        digitalWrite(in1,HIGH);
        digitalWrite(in2,LOW);
    }
    // Limit maximum PWM
    if(abs(sp)>maxPWM){
      sp = maxPWM;
    }
    else if(abs(sp)<minPWM){
      sp = minPWM;
    }
    analogWrite(pin,abs(sp));   // Write PWM to motor
}
```

The Arduino program presented in Table 7 consists of continuous reading of the data sent by the Raspberry Pi and two other main routines. The first routine is called *reachTarget()* and runs when the mobile robot has not reached the Qi charger. This routine controls the mobile robot wheel motors based on the distance and angle data from the Raspberry Pi. The second routine is called *positioning()* and runs when the mobile robot has reaches the Qi charger. When the *positioning()* routine runs, fine positioning of mobile robot is performed using the sensing coil voltage readings. Forward and backward motion, as well as the mounted positioner motion, are used in the precise coil alignment task.

Finally, there is a helper function called *motor()* that converts positive and negative PWM values to the proper motor control signals.

Together, the Pi Camera python script (Table 5) and the mobile robot driver Arduino program (Table 7) perform the two-step mobile robot alignment task presented in Figure 4. The first step moves the mobile robot a large distance towards the target, bringing the mounted sensing coils near the Qi wireless charger. The second step precisely positions the robot above the Qi wireless charger so that wireless charging can take place efficiently.

CHAPTER V

CONTROLLER DESIGN

The methodologies used to design the controller of the 2D coil positioner and mobile robot are presented in this chapter. In both systems, classical control techniques are used.

5.1 2D positioner controller

Controller design for the 2D coil positioner of Figure 31 is based primarily on two observations. First, the discrete step control of the stepper motors and the low inertia of positioner components allows for fast dynamic response with little or no overshoot. Thus, a simple control law can be used. Second, the sensing coils provide an indication of misalignment direction, but not necessarily a good measure of the misalignment amount due to the nonlinear voltage output (see Figure 11). These observations are to be confirmed through experimentation in Section 6.1. Based on these observations, a proportional (P) controller is proposed. The logic diagram in Figure 39 depicts the signal flow at each controller iteration. As mentioned in Section 2.5, the *x*- and *y*-direction loops are closed independently and each direction uses two sensing coils to measure misalignment. The controller determines the misalignment direction and send the proper step pulses to move in the direction of misalignment. A small dead band is set by a threshold voltage, v_{thresh}, to avoid limit cycling for the positioner when it reaches near alignment. Without the voltage dead band, sensing coil voltage noise could cause the positioner to jitter around the near-zero misalignment position.

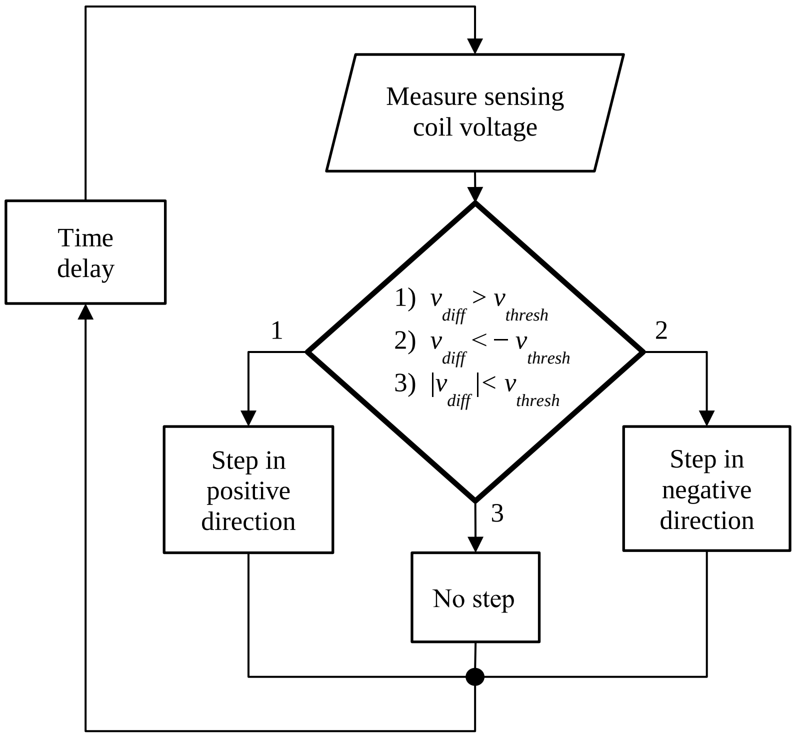

Measure sensing coil voltage
Time delay
1) $v_{diff} > v_{thresh}$
2) $v_{diff} < -v_{thresh}$
3) $|v_{diff}| < v_{thresh}$
1
2
3
Step in positive direction
Step in negative direction
No step

Figure 39. Logic diagram for IPT coil positioner automatic alignment

To achieve P control, the time delay between iterations in Figure 39 is calculated based on the sensing coil voltage difference. Greater sensing coil voltage differences result in greater stepper motor speeds. A function to calculate the time delay can take the form

$$t_{delay}(v_{diff}) = t_{d,\,max}\,(1 - |v_{diff}|/v_{max})\,, \tag{17}$$

where

t_{delay}: time delay [ms]

$t_{d,\,max}$: maximum time delay [ms]

v_{max}: maximum expected voltage difference between the sensing coils [V]

v_{diff}: voltage difference between sensing coils [V].

When calculated using (17), the maximum time delay (and slowest stepper motor speed) occurs when the sensing coil voltage difference is zero. According to the simulation results in Figure 11, low voltages occur near zero misalignment or near the sensing coil sensing range. The time delay in (17) will go to zero (and the stepper motor will reach its maximum speed) when the sensing coil voltage difference reaches its maximum value. Note that there will always be a small time delay between motor steps due to computation time. Furthermore, the maximum motor speed can be set through programming in the Arduino code using the stepper motor function *setSpeed()* and overrides any manual time delay that is too small.

A more conventional representation of the P control system is shown in Figure 40. The signal types and their units are included in the control loop, and the Arduino functions are identified.

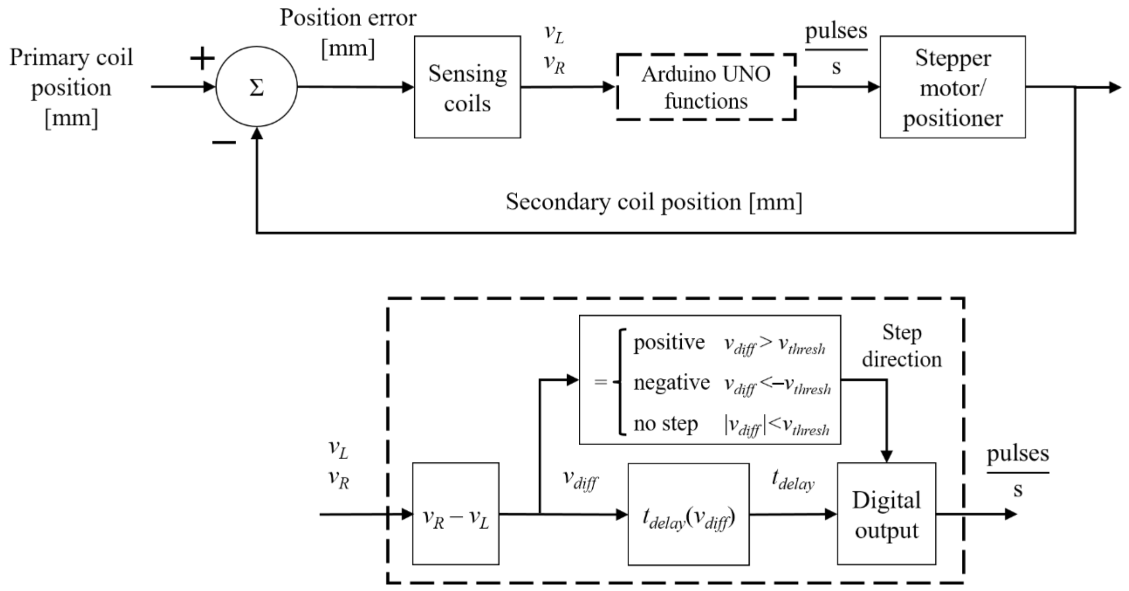

Primary coil position [mm]
+
−
Σ
Position error [mm]
Sensing coils
v_L v_R
Arduino UNO functions
pulses / s
Stepper motor/ positioner
Secondary coil position [mm]
v_L v_R
$v_R - v_L$
v_{diff}
= positive $v_{diff} > v_{thresh}$; negative $v_{diff} < -v_{thresh}$; no step $|v_{diff}| < v_{thresh}$
Step direction
$t_{delay}(v_{diff})$
t_{delay}
Digital output
pulses / s

Figure 40. Closed-loop IPT coil controller with proportional control

5.2 Mobile robot controller

A controller layout for the mobile robot application was presented in Figure 22. The goal of this closed-loop control is to allow the mobile robot to reach the target location where there is a Qi wireless charger. Closing the loop is made possible by the onboard Pi Camera that gives the mobile robot a measure of distance and heading angle error to the target. The distance and heading angle error variables are the inputs to the mobile robot controller. As previously mentioned, control signal coupling is necessary since the left and right wheels work together to drive and steer the robot. The controller form shown in Figure 41, which provides for control signal coupling, is proposed.

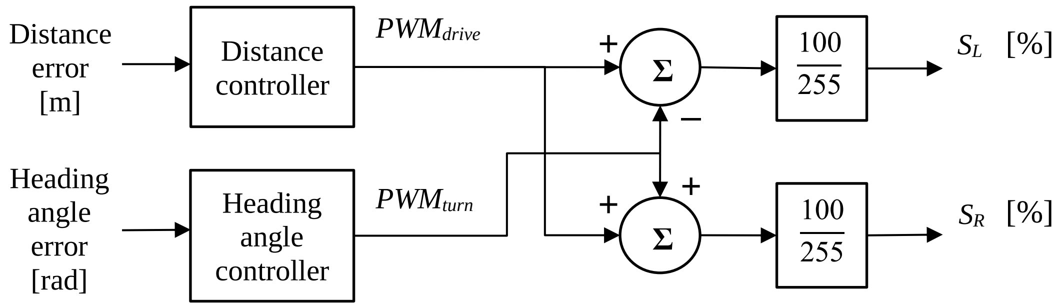


Figure 41. Proposed mobile robot controller structure and signals

In the control structure of Figure 41, the distance and angle errors are treated independently, but the control outputs are coupled. The distance error affects both the left and the right wheels equally, but the heading angle error affects the wheels in the opposite way, allowing the robot to turn when there is a heading angle error.

The output signals of the controllers in Figure 41 are Arduino PWM values that range from 0 to 255 in magnitude. Ultimately, the PWM value for each motor is converted to a duty cycle percentage. Using duty cycle control of DC motors is a common practice. The DC motor acts as a low-pass filter, and duty cycle modulation is a way of varying the effective motor driving voltage. Higher duty cycles are the equivalent of higher DC driving voltages. Note that negative PWM values and duty cycles indicate that the DC motor rotates in the opposite direction. However, negative duty cycles have no physical meaning. The Arduino program and H-bridge motor drivers are responsible for converting the PWM values to the correct duty cycle and motor current direction to control rotation direction.

To design each error controller in Figure 41, the cases of pure distance error and pure heading angle error are considered. Considering only one error at a time reduces the control system to a single-input-single-output (SISO) system. In reality, the system is a multiple-input-multiple-output (MIMO) problem. To justify the SISO approach, the goal of control is to eliminate the heading angle error much more quickly than the distance error, leaving only the SISO case of pure distance error (straight motion).

When there is only a distance error, the system's block diagram can be represented as in Figure 42. In this case, the same control signal is sent to both motors.

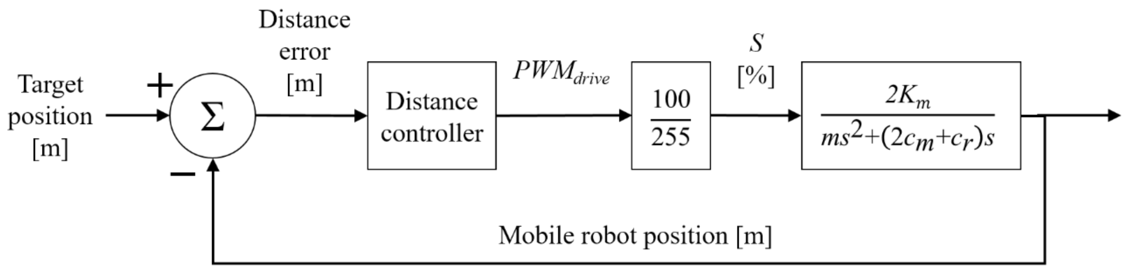


Figure 42. Equivalent robot control loop when there is only a distance error

The robot forward dynamics in Figure 42 is a Type one system if a P distance controller is used. Since the goal of the mobile robot is to reach a non-moving target, this system type is sufficient to allow zero steady state error. Let the P control gain be labeled K_p. Then, the closed-loop transfer function for pure straight motion becomes

$$\frac{200K_mK_p}{255ms^2+255(2c_m+c_r)s+200K_mK_p}. \tag{18}$$

The closed-loop transfer function (18) is stable for all positive gains K_p when there is a step reference signal, but there will be overshoot when $K_p > 255(2c_m+c_r)^2/8K_mm$. The controller gain K_p should be selected so that there is no overshoot and the mobile robot stops at the desired distance.

Using a P controller may be problematic if there are unknown or unmolded dynamics in the robot forward motion. For example, if there is a stiffness k in the robot forward dynamics, then the system with a P controller becomes Type zero and the closed-loop transfer becomes

$$\frac{200K_mK_p}{255ms^2+255(2c_m+c_r)s+(200K_mK_p+255k)} \tag{19}$$

With stiffness k included in the closed-loop transfer function (19), overshoot occurs when $K_p > 255(2c_m+c_r)^2/800K_m m - 255k/800K_m$. Furthermore, the steady-state error to a step reference input for this Type-zero system becomes $255k/(200K_mK_p+255\text{k})$. If the stiffness k is small, then it makes minor difference to the allowable gain for no overshoot and the steady-state error will still be near zero. Thus, a P controller for the mobile robot forward motion is expected to be satisfactory. The performance of this controller is simulated and tested in Sections 6.4 and 6.5.

When there is only a heading angle error, the system's block diagram can be represented as in Figure 43. In this case, the same control signal—but with opposite direction—is sent to the left and right wheel motors.

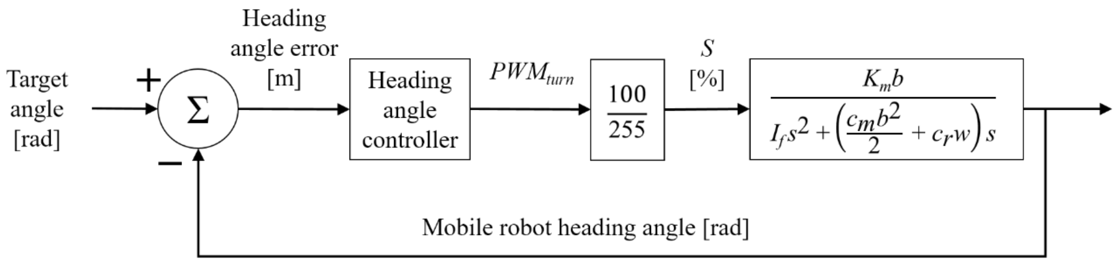


Figure 43. Equivalent robot control loop when there is only a heading angle error

The system in Figure 43 has the same structure and is the same Type as that of Figure 42 when a P controller is used. By the same arguments, a P controller is also expected to bring the heading angle error to zero. For no overshoot, the controller gain should be less than $255(c_m b^2/2+c_r w)^2/400K_m bI_f$. Once again, if there is a stiffness in the turning dynamics, the system Type will be zero and there will be a small steady state error.

For simulation and testing of the mobile robot, a P controller will be implemented for both distance and heading angle error (see Sections 6.4 and 6.5). Some steady-state error will be accepted at the end of the mobile robot trajectory, since the sensing coils will then be used to precisely position the robot over the wireless charger. To minimize errors due to the SISO approximations of Figures 42 and 43 in controller design, one performance goal will be to eliminate the heading angle error much faster than the distance error. Doing so will result in the mobile robot following a straight motion for most of its trajectory, in which case the system is essentially reduced to Figure 42. The relative speed of the distance and heading angle controllers can be assessed by comparing the control loop bandwidths of Figures 42 and 43. The selected controller gains, time response, and bandwidths are presented in Section 6.4.

CHAPTER VI

EXPERIMENTAL RESULTS

The experimental results for each thesis experiment are presented in this chapter.

6.1 Sensing-coil voltage measurements

Sensing coils were constructed as shown in Figure 28 with 10 turns per coil. The diameter of each coil was 0.5 in and the center-to-center distance between opposite sensing coils was 1.25 in. Each sensing coil was then connected to a rectifying circuit as shown in Figure 29 so that the coils' voltage amplitude could be measured by the Arduino. The 2D positioner design of Figure 31 was also created, and is shown in Figure 44.

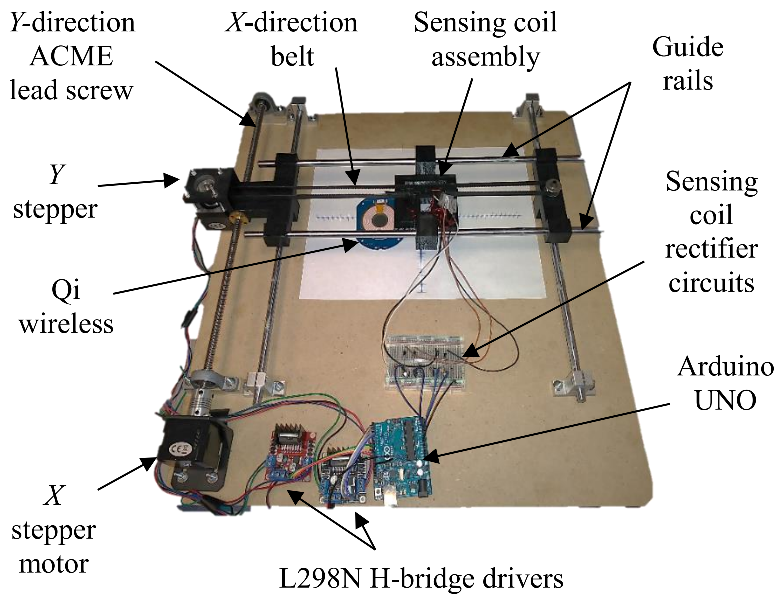


Figure 44. Experimental 2D coil positioner for sensing coil voltage measurements and automatic coil alignment experiments

As an initial test, the output voltage of the sensing coils in the presence of a primary coil was measured. The Qi wireless charger (Figure 26) was powered using a 5-V, 2-A power source and placed 1 cm directly below a sensing coil. The Qi receiver module (Figure 27) was placed between the Qi charger and the sensing coil to enable the charger operation. The open-circuit voltage of the coils with and without rectification is shown in Figure 45. Note that the frequency of the signal without rectification is about 140 kHz, which is the nominal operating frequency of the Qi wireless charger.

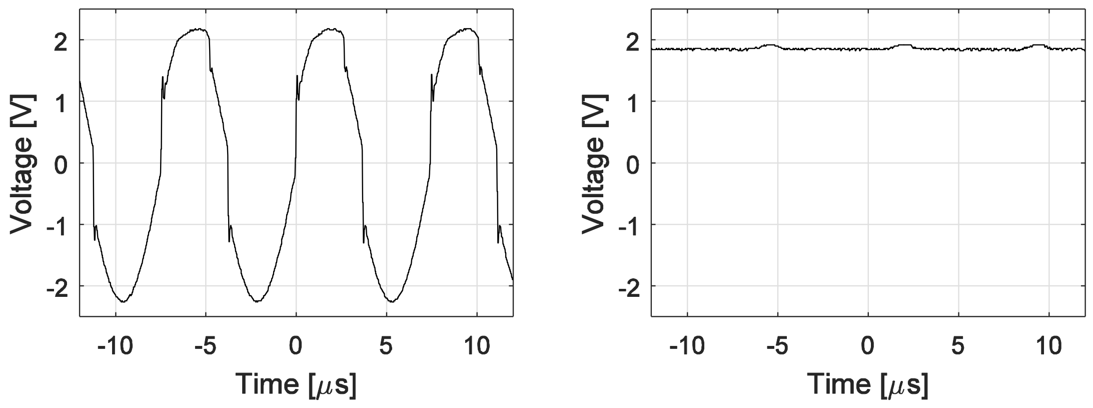


Figure 45. Sensing coil output voltage before (left) and after rectification (right)

As expected, the rectified signal in Figure 45 provides a DC measurement of the sensing coil voltage amplitude (minus a small voltage drop). A measurable coil voltage amplitude is the basis of magnetic-flux strength sensing, as described in Section 2.1. For real-time control, it is also necessary to ensure that the measurements can occur at an acceptable sampling rate. The achievable response time is dictated by the capacitor and resistor values used in the rectification circuit (10 μF and 1 kΩ, respectively). The voltage response time of the rectification circuit from 5 V to 0 V is shown in Figure 46.

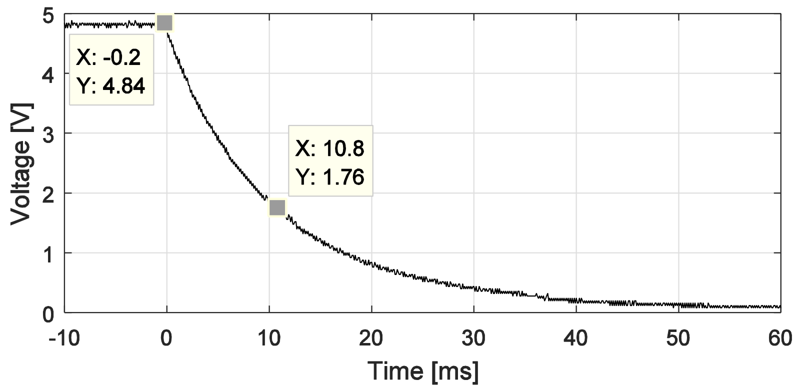


Figure 46. Voltage time response for a single sensing coil with rectification

The time response plot in Figure 46 demonstrates that the sensing coil rectifier circuits have a time constant of about 10 ms, which could also be calculated from the resistor and capacitor values. The full response time, then, is about 40 ms, allowing for a sampling rate of about 25 Hz.

With a measurable voltage amplitude of the sensing coils, the performance of sensing coils for IPT lateral misalignment detection is tested. The dual sensing coils were installed in the 2D positioner assembly of Figure 44. The sensing coils circuits, Arduino board, and stepper motors were connected according to the schematics in Figures 32 and 33. The Qi wireless charger was removed from its case, connected to power, and placed directly under the sensing coil assembly at a distance of 8 mm from the sensing coils. The Qi receiver module was placed symmetrically above the charger on top of a paper spacer of 2 mm thickness. Placing the receiver above the charger activates the transmitting circuit, indicated by a green status light on the charger. Finally, the Arduino program in Table 2 was uploaded to the Arduino board. This program performs sensing coil voltage measurements for different misalignment distances in the *x* direction of Figure 31.

With the wireless charger powered and the sensing coils centered above the charger, the Arduino program was executed and the serial monitor data was recorded. The sensing coil voltages as a function of *x* misalignment when *y* = 0 is plotted in Figure 47. The left and right coil labels refer to the sensing coil configuration illustrated in Figure 8.

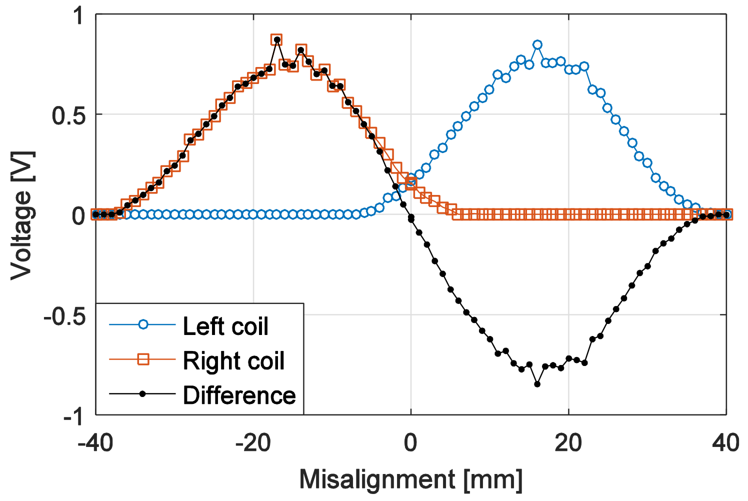


Figure 47. Sensing coil voltages versus misalignments in the *x* direction with zero misalignment in the *y* direction

The trend in Figure 47 is nearly identical to the trends from Matlab simulation (Figure 11). As misalignment increases, the voltage difference between the sensing coils increases in magnitude. At about one primary coil radius (~20 mm in the case of the Qi wireless charger), the voltage difference reaches a maximum and begins to decrease. The highest misalignment distance with a measurable voltage difference is close to one primary coil diameter (~ 40 mm). This result agrees with the predictions of Section 2.4 and supports the prediction that sensing coils can be used for measuring coil misalignment within the range of one primary coil diameter. Figure 47 also supports the claim that each sensing coil experiences an induced voltage only while it is above the primary coil radius. These results hold even though the experimental primary coil is not an ideal circular loop, as was assumed in simulations. Figure 47, also suggests that the sensing coils provide a misalignment sensing resolution of at least 1 mm. With this sensing resolution, an automatic alignment mechanism could bring the IPT coils to alignment within a few millimeters or less, which is sufficient to allow efficient power transfer [12].

To see how 2D misalignment affects the sensing coil performance, the voltage readings for *x*-direction displacements were repeated with added *y*-direction offsets (Figure 48). First, the sensing coils were offset 10 mm in the *y* direction and the voltage measurements for *x*-direction displacements were repeated.

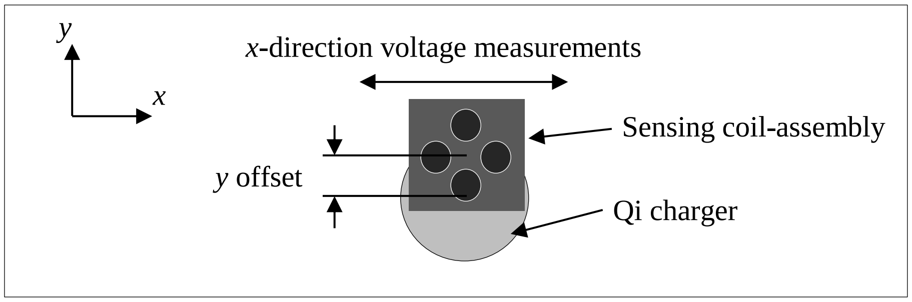


Figure 48. Offset in the *y* direction added to test 2D sensing coil misalignment voltages

The *y* misalignment of 10 mm is about half the primary coil radius. The sensing coil voltage measurements with this *y* misalignment are plotted in Figure 49. With this misalignment, the trend remains largely unchanged compared to that of Figure 47. This demonstrates that small *y*-direction misalignments do not significantly affect the *x*-direction misalignment sensing capability, increasing confidence in the 2D capability of the sensing coil configuration.

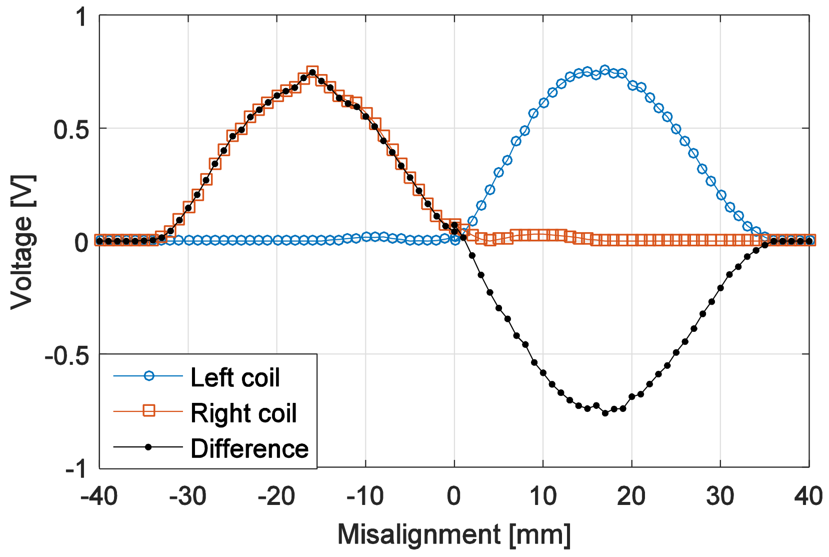


Figure 49. Sensing coil voltages for misalignments in *x* direction, with *y* offset of 10 mm

Figure 50 shows the sensing coil voltage measurements when the *y*-direction offset is increased to 15 mm. Once again, the trend remains largely unchanged. However, the voltage magnitudes are smaller compared to Figure 47 and the sensing limit is slightly decreased. Thus, the 2D performance of the sensing coils is slightly more limited in misalignment range, but it is still useful for misalignment detection.

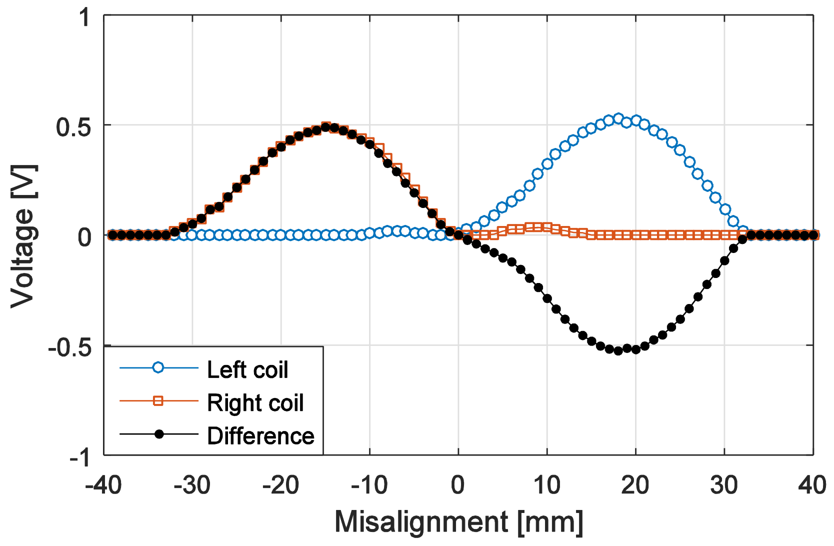


Figure 50. Sensing coil voltages for misalignments in *x* direction, with *y* offset of 15 mm

In the 2D positioner controller design of Section 5.1, any coupling between the *x* and *y* directions was ignored to simplify the analysis. However, the results in Figures 49 and 50 show that *y*-direction misalignments affect *x*-direction sensing coil voltages and vice versa. As previously stated, the main effects of 2D misalignments are decreased voltage magnitudes and decreased misalignment sensing range. Otherwise, the sensing coil voltage trends remain largely unaffected. With the sensing coil configuration of Figure 8, the range of detectable 2D misalignment distances will always be slightly smaller than that of the 1D misalignment case. The key to any misalignment detection is to ensure that at least one sensing coil remains within the area directly above the primary coil. The result of ignoring coupling between the two directions is that the 2D positioner may, in some instances, not follow a direct path to alignment. Figure 18 predicts some of the alignment trajectories under several 2D misalignment conditions.

6.2 Automatic 2D positioner performance

In the second thesis experiment, the automatic alignment capability of the 2D positioner (Figure 44) was tested. The sensing coil voltages were implemented in the control scheme in Figure 40 by using the Arduino programs in Table 3.The system was wired according to the schematics shown in Figures 32 and 33. Before the program was executed, the Qi wireless charger was centered under the sensing coils at a distance of 8 mm. The Qi receiver module was placed between the charger and the sensing coils with a paper spacer of 2 mm above the charger. The presence of the receiver module enables the wireless charger, as indicated by a green indicator light. The Arduino program was then executed. This program displaces the sensing coils in the specified *x* and *y* directions of Figure 31 and then activates the controller to measure the system response.

First, 1D misalignment reduction was tested. The closed-loop controller described in Section 5.1 was used. The values used for time delay calculation (17) were $t_{d,max}$=20 ms and v_{max}=0.75 V. The value v_{max} was selected based on the voltage results of Figure 47 and the $t_{d,max}$ value was selected arbitrarily to set the slowest stepper motor speed. The threshold voltage (v_{thresh}) used to prevent limit cycling of the positioner was chosen to be 0.05 V. This value was experimentally determined by trial-and-error so that the positioner did not jitter around the aligned position. The time response for several initial misalignments in the *x* direction are plotted in Figure 51. The error lines ±1 mm bound the steady state errors of all the trials, and the effects of the P control can be seen in the changing slope of the *x* position error in time.

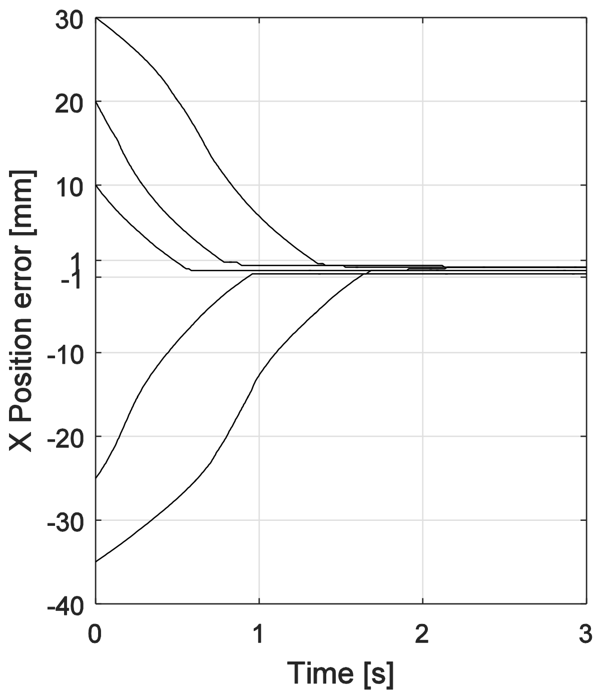


Figure 51. Automatic positioner performance for several initial misalignments in the *x* direction

Figure 51 shows that the closed-loop controller reduces the coil *x* misalignment to less than 1 mm in a response time of less than 2 s. The effect of using speed control is a fast motor movement from about 10- to 20-mm *x* misalignment, but slower movement otherwise. This behavior is a result of the voltage trend of Figure 47. As expected, the controller reduced the misalignment quickly without significant overshoot. This result was possible due to the 2D positioner dynamics (step control and low system inertia). In positioner systems that are more susceptible to overshoot, such as may be the case for systems that use low-friction DC motors, the $t_{d,max}$ value used in (17) can be increased to give larger time delay values and a slower closed-loop system response.

Near 40 mm initial *x* misalignment (not shown in Figure 51), automatic control action fails because the sensing coils are no longer within their sensing range. From Figure 47, it is evident that the farthest misalignment that can be detected is about 35 mm; larger misalignments leave the sensing coils outside of the significant magnetic flux region and result in no induced voltages. To complement the sensing range, a positioning system could scan the area until the sensing coils detect the primary coil. Otherwise, an alternative sensor would be required to bring the sensing coils near the primary coil.

The time response for 2D misalignments was also tested. To do this, initial misalignments in both *x* and *y* direction are imposed on the positioning system. The time response for five initial misalignments are plotted in Figure 52, and the resulting *xy*-plane trajectories for the same five trials are plotted in Figure 53.

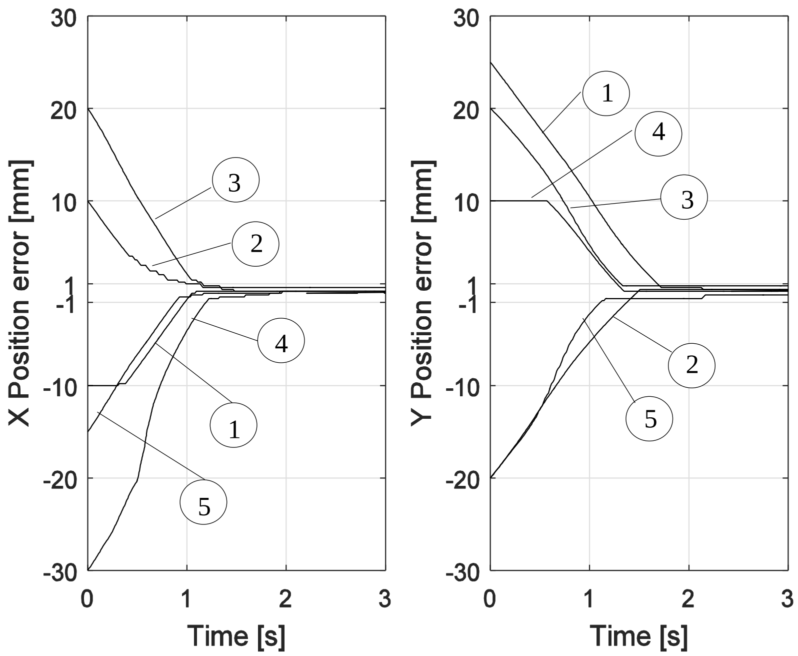


Figure 52. Automatic positioner response for several 2D initial misalignments

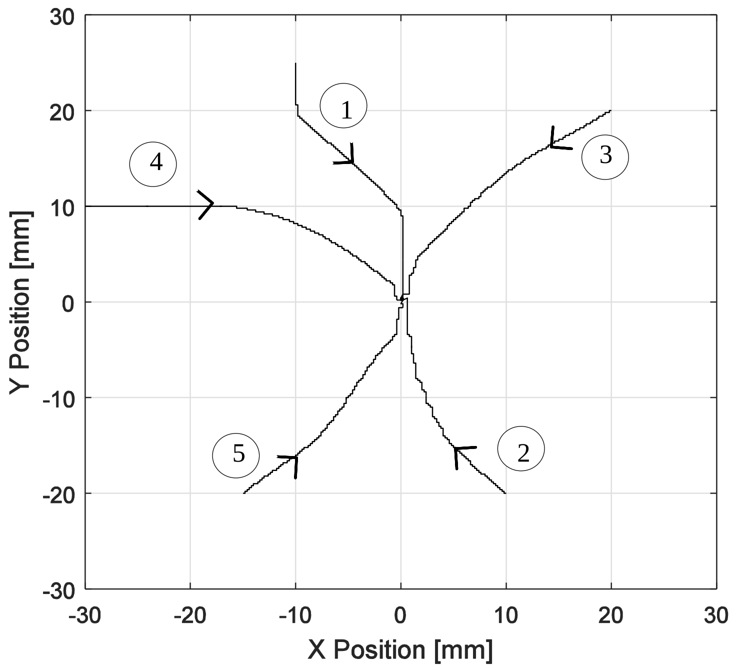


Figure 53. Positioner trajectory for correction of several initial misalignment in the *x* and *y* directions (same trials plotted in Figure 52)

Figures 52 and 53 show that in the 2D case the controller is still able to reduce misalignments to 1 mm or less in each direction. As a result, the maximum lateral error that can be expected at the end of the positioning trajectory is about $\sqrt{2}$ mm or less. Once again, this lateral position error is small enough to allows for efficient power transfer [12].

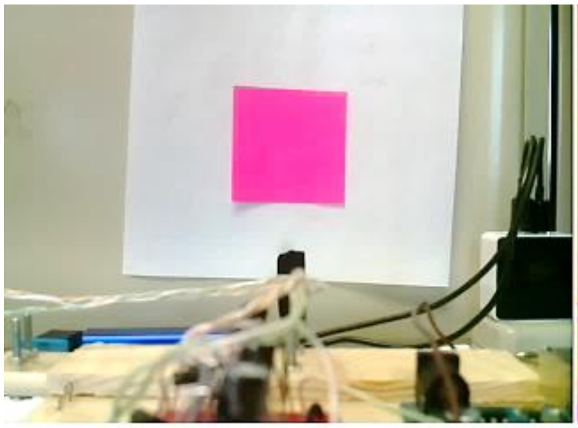
Two-dimensional automatic positioning failed when the initial misalignment was increased to 22.5 mm in *both* the *x* and *y* directions. This kind of misalignment represents the worst-case scenario for misalignment (Scenario (b) in Figure 18) where all sensing coils go beyond the region of appreciable magnetic flux. This result highlights the dependence of misalignment sensing on the misalignment direction and sensing coil orientation. Recall that the sensing range extended to nearly 35 mm in the 1D case (Figure

47). Furthermore, note that the alignment trajectory for some of the trials in Figure 53 were not straight. Scenario (c) in Figure 18 occurs in trials 1 and 4. In other trials, the misalignment in one direction is reduced faster than the other, also resulting in a non-straight alignment path. The alignment path does not affect the final position error and is a consequence of using independent *x* and *y* controllers.

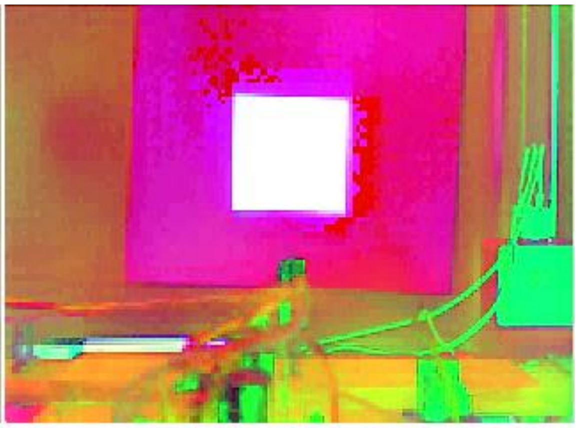
6.3 Mobile robot camera tracking

The third thesis experiment tests the application of automatic IPT alignment to the mobile robot wireless charging scenario first presented in Figure 4. First, the mobile robot was given the ability to detect a colored target that would mark the location of a Qi wireless charger. The Pi camera and Raspberry Pi were installed on the mobile robot as prescribed in Section 3.5. The Raspberry Pi was connected to the Arduino according to the schematic of Figure 38. The image filtering is the first step of target-tracking algorithm that allows for a calculation of distance and heading angle error to the target as seen by the mobile robot.

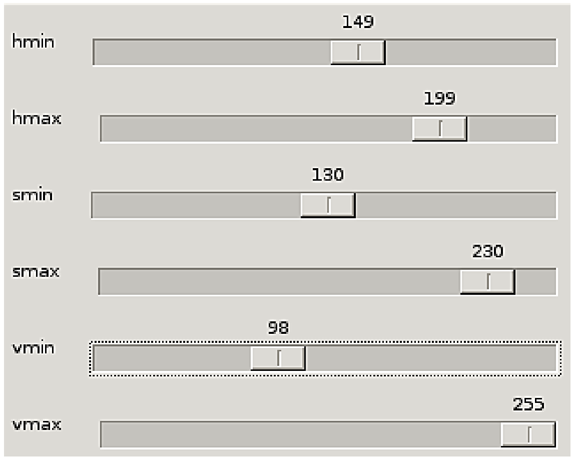
To test the image filtering capability of the Pi Camera, the Python script presented in Table 6 was loaded and executed in the Raspberry Pi. A pink Post-it note (paper size 3 in by 3 in) was used as the colored target. This color was chosen because it stands out against most objects in common environments. With the Pi Camera facing the target and a steady light source (room fluorescent lights and other ambient light), HSV thresholds values were adjusted until everything but the target was filtered out of the live Pi Camera images (Figure 54).



(a)

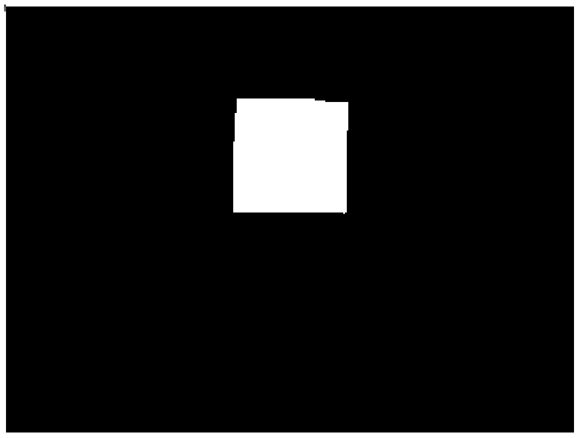


(b)



hmin 149
hmax 199
smin 130
smax 230
vmin 98
vmax 255

(c)



(d)

Figure 54. HSV filtering of camera image frame for target detection; (a) Original RGB image (b) HSV image (c) HSV threshold value selection window (d) Filtered image

Figure 54 shows that in the filtered image only the colored target remains. Thus, the target was successfully highlighted using the following HSV threshold values:

- Hue: 150 to 200
- Saturation: 130 to 230
- Value: 100 to 255

These threshold values allowed for successful image filtering even when the ambient lighting conditions changed slightly. This result is important because it makes the camera sensing more robust to environmental changes.

With suitable HSV threshold values for image filtering identified, the target-tracking script of Table 4 was then executed on the Raspberry Pi to test the area and position calculations of the script. For this test, no power was provided to the mobile robot motors so that it did not move. The tracked image displayed as shown in Figure 55.

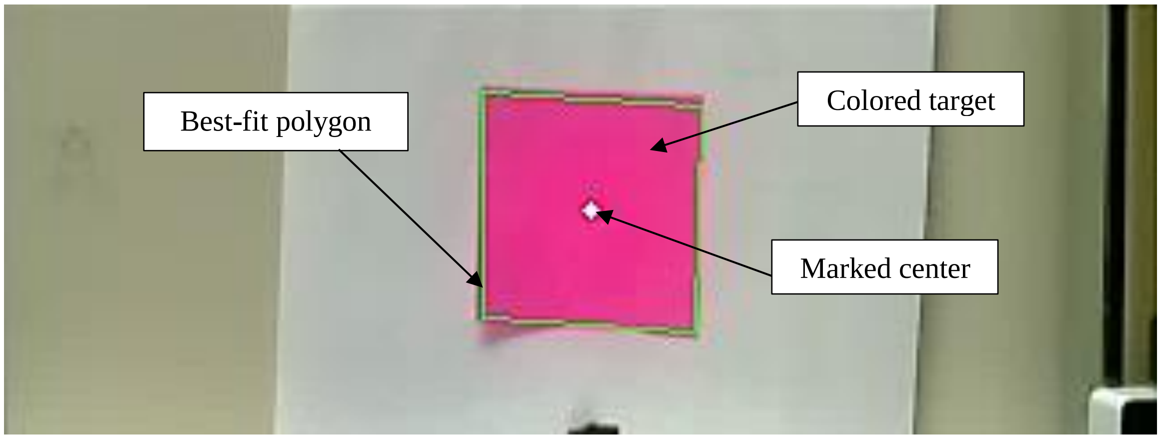


Figure 55. Tracked target display with polygon contour and center marked

The tracked image in Figure 55 demonstrates that the target color is successfully found and outlined using the Raspberry Pi script in Table 4. Furthermore, the center of the target is identified. The area of the target in Figure 55 is calculated to be about 60 by 60 pixels and the horizontal position of the target center is 157 pixels. These results make sense since the image resolution is 320 by 120 pixels. To further demonstrate the ability of the Raspberry Pi script to locate the target in the image frame, a foreign object is introduced into the field of view in Figure 56.

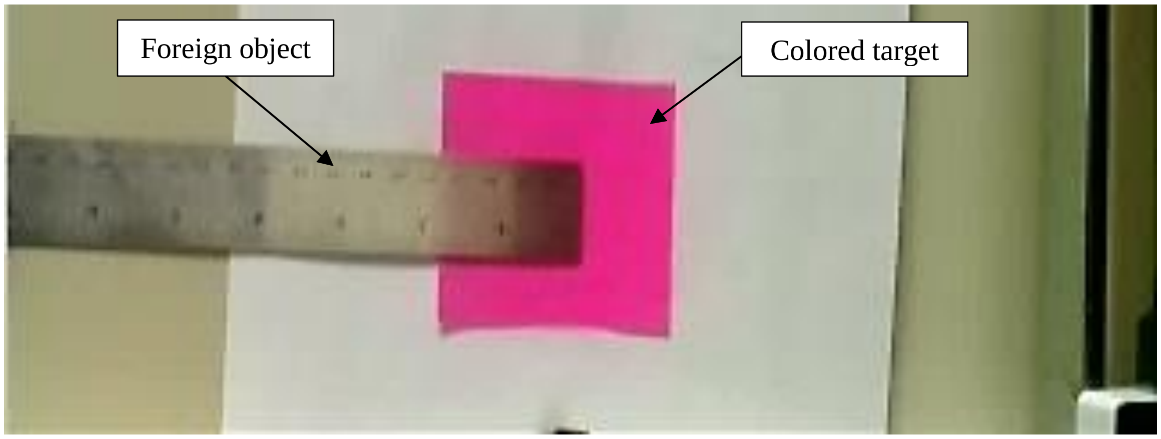


Figure 56. Foreign object interference in target detection

When a foreign object obstructs the camera view, the pink target no longer fits the target profile specified in the image-processing script (the target must be a four-sided polygon) and it is no longer identified as the target. This demonstrates that the target-tracking python script detects the colored target faithfully.

To close the control loop in Figure 22, the Pi Camera and image-processing script output a measure of distance and angle error to the target. The distance is calculated from the size of the colored target in the image. The angle error is calculated from the horizontal position of the target in the image.

A calibration procedure was performed to relate the distance to the target and the target pixel area in the image. A plot of target pixel area for various measured distances is shown in Figure 57. Also included in the plot is a line that fits the results.

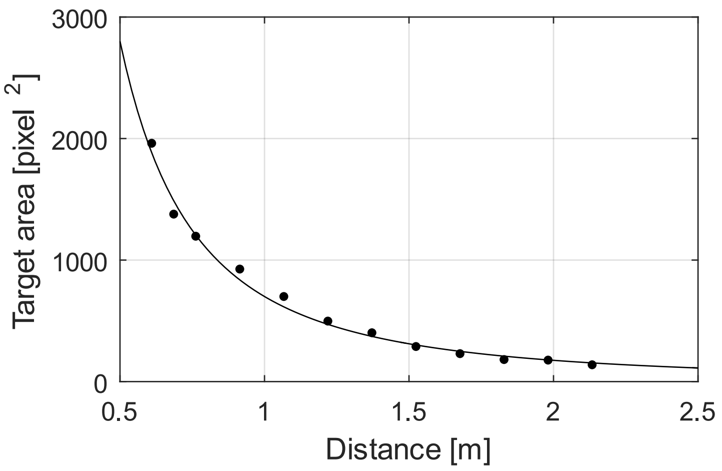


Figure 57. Pixel area of target for known distances with line fit

As expected, larger distances correlate with smaller target pixel area. The fit line can serve to model this result. The equation of the fir line is rearranged to solve for distance and give the expression

$$Dist = \sqrt{\frac{700}{A_{px}}} \ , \tag{20}$$

where

Dist: distance to target [m]

A_{px}: target area [pixel^2].

Equation (20) is implemented in the camera sensor script (Table 4) to calculate distance to the target. The heading angle error to the target is calculated similarly, but based on the position of the target in the image. The total field of view for the Pi Camera is 53° in the horizontal direction. The horizontal position of the target in the image can be used to estimate the heading angle error using the equation

$$\text{Heading angle error} = \left(\frac{53}{2} - \frac{53X}{\text{image width}}\right), \qquad (21)$$

where X is the horizontal pixel location of the target center. Equation (21) maps the horizontal pixel location X of the target from 26.5° at the left edge of the image to −26.5° on the right edge of the image. When the target is straight ahead of the robot, it lies at half the image width and the heading angle error given by (21) is zero.

With (20) and (21) implemented in the image tracking script (Table 4) the camera approximately performs the sensor functions given in Figure 24. The target-tracking script captures images, process them, and calculates distance and heading angle errors at about 30 frames per second. The framerate was measured by adding a line of code to the python script which output the time of each new frame capture.

6.4 Mobile robot performance simulation

The third experiment of this thesis tests the automatic control of a mobile robot to align it with a wireless charger. This scenario is presented in Figure 4. Initially, the mobile robot approaches the wireless charger location, which is marked by a colored target. In this section, the mobile robot parameters are experimentally determined and the mobile robot target approach is simulated with the control law presented in Section 5.2.

The dynamics of the differential drive mobile robot are presented in (16). Now, the parameters in the dynamical equations are estimated. The mass and dimensions of the mobile robot are measured directly. The robot moment of inertia, motor amplification constants, and damping coefficients are estimated by matching the simulation results to experimental observations.

First, a control signal of 78% duty cycle (Arduino PWM value of 200) was given to each wheel motor for two seconds. The robot, which started at rest, traveled straight for about 0.82 m (2.7 ft). The robot stopped almost immediately after the two-second interval was over. In a second observation, the left wheel motor was given a control signal of 78% duty cycle in one direction and the right wheel a control signal of 78% in the other direction for two seconds. These control signals caused the robot to turn in place a total of 9/4 π rad. Once again, the robot stopped almost immediately after the two second interval. The straight and turning motion observations, compared to simulation results from Matlab, allowed for the robot parameters to be estimated. All of the parameter estimates and the methods of estimation are summarized in Table 8. These estimates are intended only to allow for basic controller validation using Matlab/Simulink.

Table 8. Mobile robot dynamic parameters used for simulation

Category	**Method**	**Value**
Measured	Ruler or scale measurement	m = 0.6 kg b = 0.15 m w = 0.12 m
Estimated based on experimental observations	Rotational motion observation	I_f = 0.02 kg·m^2
	Robot weight balance	d = 0.07 m
	Motor torque and PWM signal	K_m = 0.0255 N/% duty
	Straight and rotational motion observations	c_m = 4.8 N·s/m c_r = 0.25 N·s/m

In Table 8, the motor gain K_m was estimated by linearly mapping the duty cycle signal (in percent) to the advertised motor torque capability. The robot center of mass location (d) was estimated by finding $\boldsymbol{i}$ coordinate which would balance the robot.

Figure 58 shows the simulation results when the estimated robot parameters of Table 8 are used in the dynamic equations. The plots represent the robot response of pure rotation and pure straight motion—the same kinds of motions that were observed and measured for parameter estimation. The simulation results of Figure 58 match closely with the observed responses of the mobile robot, suggesting that the estimated robot parameters are acceptable for simulation to some level.

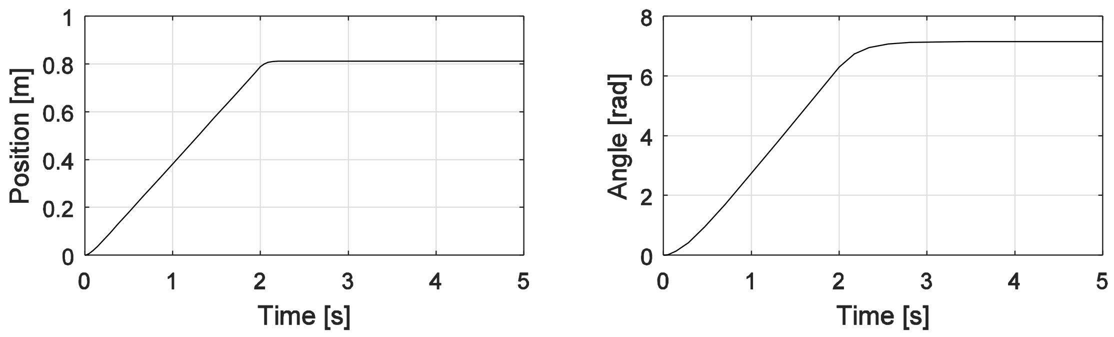


Figure 58. Simulation of mobile robot response for pure straight motion (left) and pure rotation (right) with two-second force inputs

With the robot parameters estimated, the mobile robot controller presented in Section 5.2 were simulated using the Matlab/Simulink setup shown in Figure 22. From controller analysis of Section 5.2, the proportional gains should not exceed 2021 PWM-value per meter (793 % duty/m) for the drive controller or 58.8 PWM-value per rad (23 % duty/rad) for turning to avoid overshoot. The values of 250 PWM-value per meter (98 % duty/m) and 573 PWM-value per radian (225 % duty/rad) were selected for the drive and steering controller, respectively, so that a pure distance error of 1 meter gives a motor control signal of 98% duty and a pure angle error of 25° (0.44 rad) gives a motor control signal of 99% duty. The maximum angle error which can be sensed by the Pi Camera is about 26°. These P gains give bandwidths of 0.52 rad/s and 9.43 rad/s for the loops in Figures 42 and 43, respectively. This means that the angle error response is much faster than the distance error response. For controller simulation, the target position is set to 1 m in the ***I*** direction and 0.3 m in the ***J*** direction and the mobile robot begins at the origin facing the ***I*** direction. The distance offset is set to 0.5 m. The simulation with these positons and controller gains gives the responses shown in Figure 59.

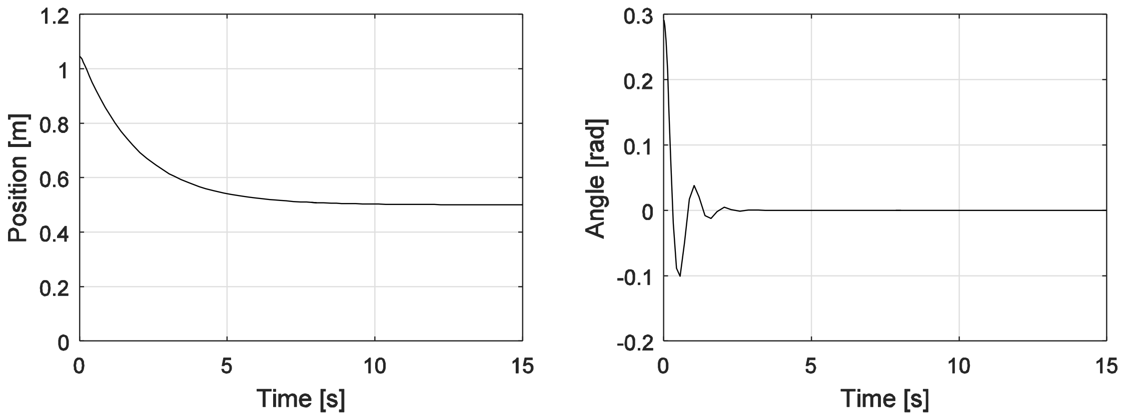


Figure 59. Proportional controller simulation response

Figure 59 shows that the P controllers allow the distance error to reach the offset of 0.5 m in about 10 s. The heading angle error is brought to zero in about 3 s—much faster than the distance error. Therefore, most the trajectory is straight motion and the SISO approximations used in controller design are justified.

The controller gains used in simulation are satisfactory and were implemented in the actual mobile robot system with the expectation of similar performance. However, upon experimenting with the mobile robot, it was found that there is a dead band region for control of the motors; motor drive signals with a magnitude lower than about 15 % duty cycle produce no motion due to friction in the motor gearbox. To address this, all signals that fell under 15 % duty cycle were rounded up to 15 %. The results of this are described in the overall system performance discussion that follows.

6.5 Overall mobile robot system performance

The final thesis experiment replicates the mobile robot wireless charging scenario presented in Figure 4. This experiment tests whether the mobile robot (Figure 34) can approach and position itself above a Qi wireless charger precisely. To accomplish this, the target-tracking script of Table 4 is executed on the Raspberry Pi, while the program in Table 7 runs on the Arduino board. In the Arduino program, the distance set point of 0.5 m is defined, and the controller gains simulated in Section 6.4 are implemented.

First, the capability of the mobile robot to reach the colored target is tested. According to the threshold values set in the Arduino code (Table 7), the mobile moves until the angle error reaches zero and the distance error reaches less than 0.01 m. Figures 60 and 61 show the distance and heading angle error of the mobile robot as it approached the target in five experimental trials. At the start of each trial, the mobile robot was placed directly ahead of the target but not facing it directly. Varying the initial placement of the robot gave a variety of initial distance and heading angle errors.

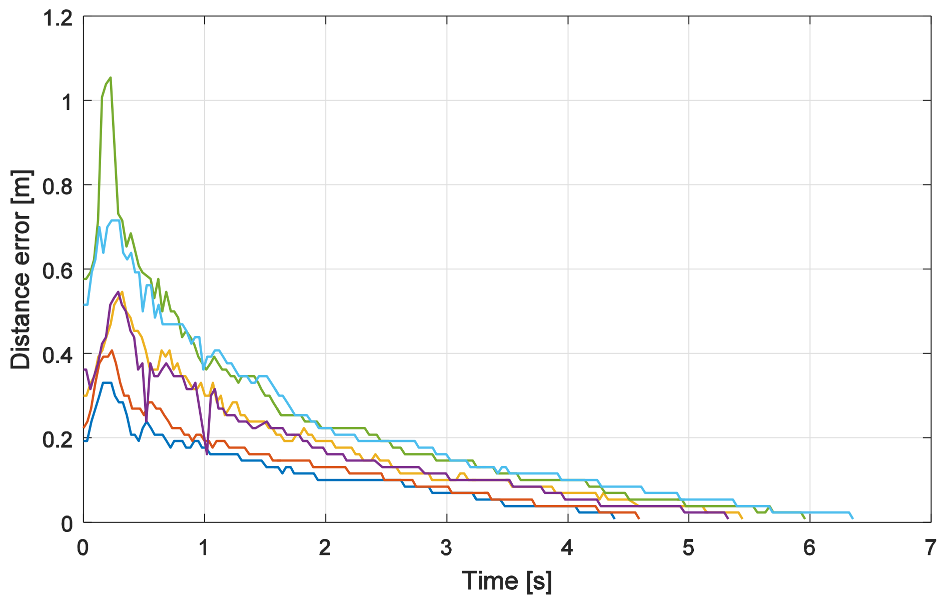


Figure 60. Distance error versus time for several experiment trials of the mobile robot target approach

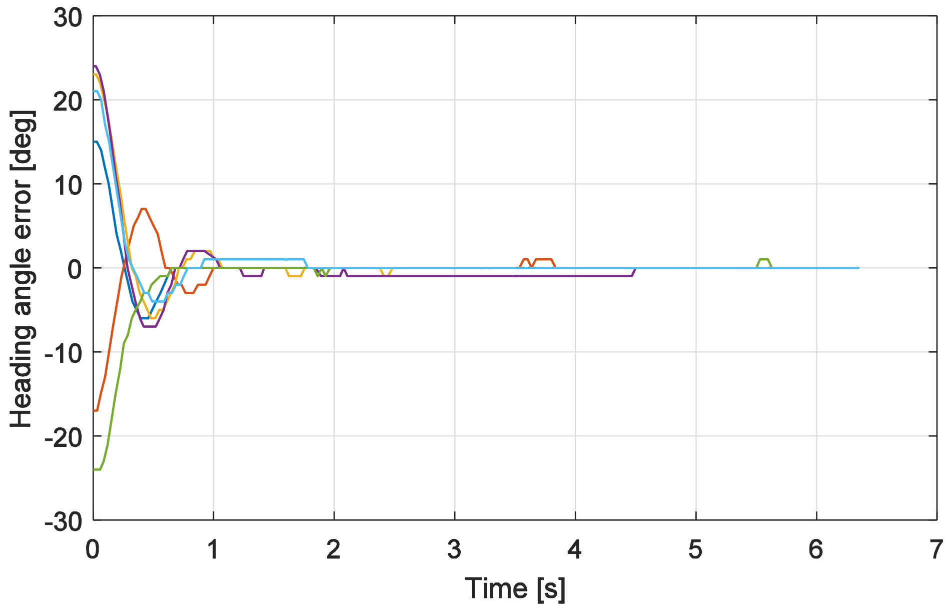


Figure 61. Heading angle error versus time for several experiment trials of the mobile robot target approach

The distance and heading angle errors plotted in Figures 60 and 61 demonstrate similar time responses as the simulations in Figure 59. Specifically, there is overshoot in the heading angle error but it is eliminated much faster than the distance error. Therefore, most the robot movement is straight motion. The angle and heading angle errors are eventually brought within the accepted limits and the robot stops.

Given the possible errors in distance and angle calculations due to image noise, pixel resolution, and random Pi Camera variations, the mobile robot is not precisely aligned with the Qi charger at the end of the experimental trials presented in Figures 60 and 61. At the end the trials, the sensing coils give the voltage measurements shown in Figure 62.

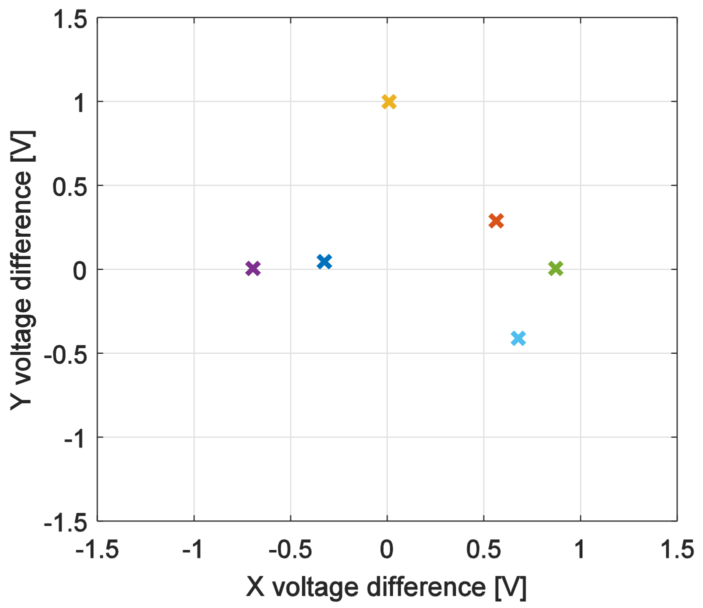


Figure 62. Initial sensing coil voltage difference upon completion of camera-assisted target approach for mobile robot

The sensing coil voltages in Figure 62, along with the sensing coil voltage plots of Section 6.1, show that there is IPT misalignment in the order of centimeters after the mobile robot has reached the Qi wireless charger. In other words, the Pi Camera alone is unable to position the mobile robot above the wireless charger precisely, and this will result in wireless charging inefficiencies. However, the final misalignment of a few centimeters allows the sensing coils to detect the wireless charger's magnetic field. With the mobile robot in proximity of the Qi charger, the voltage output from the sensing coils can be used to precisely positon robot above the wireless charger without further relying on the camera sensor. This control action is demonstrated in the voltage difference versus time plot (Figure 63) for one of the experimental trials is presented in Figure 62. The onboard positioner is used to reduce the *x*-direction error and the robot motor wheels are used to reduce the *y*-direction error.

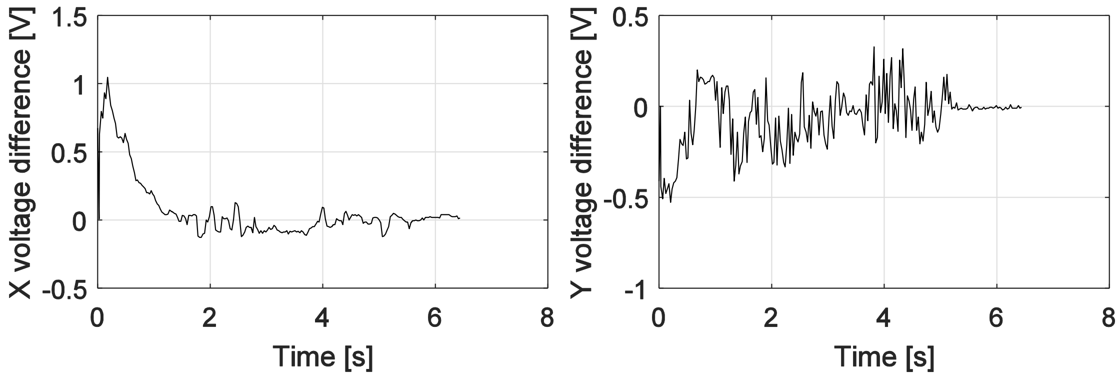


Figure 63. Sensing coil voltage difference in *x* and *y* directions through time with the use of closed-loop coil positioning

Similar results as those in Figure 63 were obtained for the other robot experiment trials. Much like the results when using a 2D coil positioner (Section 6.2), Figure 63 shows that the mobile robot-mounted sensing coils and positioning system can reduce IPT misalignments so that the voltage difference in the sensing coils falls to less than 0.1 V. At this low voltage difference, the IPT system is aligned within a few millimeters (see Figure 47). This precise alignment is more than could be offered using the Pi Camera alone (see Figure 62). Without the control capability provided by the sensing coils and mounted 1D positioner, the misalignment of a few centimeters would go uncorrected. Misalignment of a few centimeters would either decrease wireless charging efficiency or not allow wireless charging to take place at all.

The overall system performance is deemed satisfactory since the sensing coil and positioning system were successful in correcting lateral misalignments in the mobile robot wireless charging experiments. However, the Pi Camera was still required to bring the mobile robot within centimeters of the wireless charger. This means that, although the sensing coils are useful for small misalignment detection and correction—within millimeters in the case of the Qi wireless phone charger—the additional Pi Camera was required to bring the mobile robot close to the wireless charger in the first place.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

Through electromagnetic principles, simulation, and experimentation, the idea of using sensing coils for lateral misalignment reduction in an IPT system was demonstrated to be successful. The sensing-coil configuration presented in this thesis is sufficient for detecting misalignments in the two lateral directions, although the non-linear signal output of the sensing coils is not quite useful for calculating the magnitudes of the misalignments. The sensing coils presented provide the advantages of being non-intrusive to the wireless charging system, occupying a small area, having a simple operating circuit, and being contactless. One limitation for their usage is that at least one sensing coils must remain within approximately the primary coil radius for misalignment detection to occur. This is due to the magnetic-field distribution created by the circular primary coil and may warrant the use of additional application-specific sensors to address larger misalignments.

The three tasks of determining sensing coil performance, demonstrating automatic positioning on an experimental 2D positioning system, and evaluating performance on a mobile robot wireless charging application were accomplished. With the Qi wireless charging system used, the sensing coils could be used to detect misalignments as small as 1 mm or about 2% of the primary coil diameter. With this misalignment detection capability, closed-loop control automatically reduced misalignment to less than 1 mm in each lateral direction, which allowed efficient power transfer to occur. Similar results were obtained when the sensing coils were implemented in the mobile robot wireless charging experiment. Although the sensing coils are useful for precise misalignment detection, they are unable to sense larger misalignments. In the case of the mobile robot, the limited sensing coil range prompted the use of a Pi Camera that allowed the robot to approach the wireless charger location autonomously.

A larger-scale IPT wireless charging system with an axisymmetric primary coil can also benefit from the use of sensing coils for lateral misalignment detection and automatic coil positioning since the working principle and analysis presented in this thesis can also apply to that system. Future work that can be done in this area includes testing sensing coils for misalignment detection on large-scale IPT systems, such as electric vehicle chargers, or even in systems with different coil geometries. It may also be possible to expand the use of sensing coils to detect other types of IPT misalignment, such as angular misalignment, that also reduce charging efficiency. The work presented in this thesis can serve as a foundation for modeling, simulating, and experimental design for expanding the use of sensing coils for IPT misalignment detection in general. The mobile robot analysis and experimentation in this thesis also contribute to the understanding of vision-based autonomous navigation, which has applications outside of robotics—including autonomous vehicle control. This area of study can further be complemented by performing advanced controller design to increase the reliability, robustness, and efficiency of autonomous vehicles in vision-based control.

REFERENCES

[1] C. K. Lee, W. X. Zhong, and S. Y. R. Hui, “Recent progress in mid-range wireless power transfer,” in *Proceedings of 2012 IEEE Energy Conversion Congress and Exposition (ECCE)*, Raleigh, NC, Sep. 2012, pp. 3819–3824.

[2] F. T. Ulaby and U. Ravaioli, “Maxwell’s equations for time-varying fields,” in *Fundamentals of Applied Electromagnetics*, 7th ed. Upper Saddle River, NJ: Pearson, 2015, ch. 6, sec. 1, pp. 282–283.

[3] W. Stanley, “Alternating-current development in America,” *Journal of the Franklin Institute*, vol. 173, no. 6, pp. 561–580, Jun. 1912.

[4] Wireless Power Consortium. (2016, November 30). *Get the Specs on Qi* [Online]. Available: https://www.wirelesspowerconsortium.com/developers/specification.html.

[5] C. R. Valenta and G. D. Durgin, “Harvesting wireless power: survey of energy-harvester conversion efficiency in far-field, wireless power transfer systems,” *IEEE Microwave Magazine*, vol. 15, no. 4, pp. 108–120, Jun. 2014.

[6] M. Kline, I. Izyumin, B. Boser, and S. Sanders, “Capacitive power transfer for contactless charging,” in *Proceedings of 2011 Twenty-Sixth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, Fort Worth, TX, 2011, pp. 1398–1404.

[7] M. G. L. Roes, J. L. Duarte, M. A. M. Hendrix, and E. A. Lomonova, “Acoustic energy transfer: A review,” *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 242–248, Jan. 2013.

[8] D. Mishra, S. De, and K. R. Chowdhury, “Charging time characterization for wireless RF energy transfer,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 4, pp. 362–366, Apr. 2015.

[9] S. Y. R. Hui, W. Zhong, and C. K. Lee, “A critical review of recent progress in mid-range wireless power transfer,” *IEEE Transactions on Power Electronics*, vol. 29, no. 9, pp. 4500–4511, Sep. 2014.

[10] X. d. T. García, J. Vázquez, and P. Roncero-Sánchez, “Design, implementation issues and performance of an inductive power transfer system for electric vehicle chargers with series–series compensation,” *IET Power Electronics*, vol. 8, no. 10, pp. 1920–1930, Oct. 2015.

[11] T. P. Duong and J. Lee, “A dynamically adaptable impedance-matching system for midrange wireless power transfer with misalignment,” *Energies*, vol. 8, no. 8, pp. 7593–7617, Jul. 2015.

[12] Y. Gao, A. Ginart, K. B. Farley, and Z. T. H. Tse, “Misalignment effect on efficiency of wireless power transfer for electric vehicles,” in *Proceedings of IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 3526−3528, Mar. 2016.

[13] R. W. Carlson and B. Normann, “Test results of the PLUGLESS^{TM} inductive charging system from Evatran Group, Inc.,” *SAE International Journal of Alternative Powertrains,* vol. 3, no. 1, pp. 64−71, May 2014.

[14] Z. Dang, Y. Cao, and J. A. A. Qahouq, "Reconfigurable magnetic resonance-coupled wireless power transfer system," *IEEE Transactions on Power Electronics*, vol. 30, no. 11, pp. 6057–6069, Nov. 2015.

[15] B. H. Waters, B. J. Mahoney, G. Lee, and J. R. Smith, "Optimal coil size ratios for wireless power transfer applications," in *Proceedings of 2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Melbourne VIC, Jun. 2014, pp. 2045–2048.

[16] K. Lee, Z. Pantic, and S. M. Lukic, "Reflexive field containment in dynamic inductive power transfer systems," *IEEE Transactions on Power Electronics*, vol. 29, no. 9, pp. 4592–4602, Sep. 2014.

[17] S. A. Mirbozorgi, M. Sawan, and B. Gosselin, "Multicoil resonance-based parallel array for smart wireless power delivery," in *Proceedings of 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, Jul. 2013, pp. 751–754.

[18] S. A. Mirbozorgi, H. Bahrami, M. Sawan, and B. Gosselin, "A smart multicoil inductively coupled array for wireless power transmission," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6061–6070, Nov. 2014.

[19] W. X. Zhong, X. Liu, and S. Y. R. Hui, "A novel single-layer winding array and receiver coil structure for contactless battery charging systems with free-positioning and localized charging features," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 9, pp. 4136–4144, Sep. 2011.

[20] J. P. W. Chow, N. Chen, H. S. H. Chung, and L. L. H. Chan, “An investigation into the use of orthogonal winding in loosely coupled link for improving power transfer efficiency under coil misalignment,” *IEEE Transactions on Power Electronics*, vol. 30, no. 10, pp. 5632–5649, Oct. 2015.

[21] I. Mayordomo, T. Dräger, P. Spies, J. Bernhard, and A. Pflaum, “An overview of technical challenges and advances of inductive wireless power transmission,” *Proceedings of the IEEE*, vol. 101, no. 6, pp. 1302–1311, Jun. 2013.

[22] S. Lee, J. Huh, C. Park, N. S. Choi, G. H. Cho, and C. T. Rim, “On-line electric vehicle using inductive power transfer system,” in *Proceedings of 2010 IEEE Energy Conversion Congress and Exposition*, Atlanta, GA, Sep. 2010, pp. 1598–1601.

[23] J. Juh and C. Rim, “KAIST wireless electric vehicles - OLEV,” in *Proceedings of the 1st International Electric Vehicle Technology Conference*, May 2011 ©SAE International and ©Society of Automotive Engineering of Japan, inc.

[24] K. Hwang, J. Park, D. Kim, H. H. Park, J.H. Kwon, S. I. Kwak, and S. Ahn, “Autonomous coil alignment system using fuzzy steering control for electric vehicles with dynamic wireless charging,” *Mathematical Problems in Engineering*, vol. 2015, Nov. 2015.

[25] Y. Gao, A. A. Oliveira, K. B. Farley, and Z. T. H. Tse, “Magnetic alignment using existing charging facility in wireless EV chargers,” *Journal of Sensors*, vol. 2016, Dec. 2015.

[26] Evatran Group Inc. (2016, November 30). *Meet the PLUGLESS L2* [Online]. Available: https://www.pluglesspower.com/.

[27] S. A. Mirbozorgi, H. Bahrami, M. Sawan, and B. Gosselin, “A smart cage with uniform wireless power distribution in 3D for enabling long-term experiments with freely moving animals,” *IEEE Transactions on Biomedical Circuits and Systems,* vol. 10, no. 2, pp. 424–434, Apr. 2016.

[28] K. Eom, J. Jeong, T. H. Lee, J. Kim, J. Kim, S. E. Lee, and S. J. Kim, “A wireless power transmission system for implantable devices in freely moving rodents,” *Medical & Biological Engineering & Computing*, vol. 52, no. 8, pp. 639–651, Jun. 2014.

[29] A. E. Rendon-Nava, J. A. Diaz-Mendez, L. Nino-de-Rivera, W. Calleja-Arriaga, F. Gil-Carrasco, and D. Diaz-Carrasco, “Study of the effect of distance and misalignment between magnetically coupled coils for wireless power transfer in intraocular pressure measurement,” *The Scientific World Journal*, vol. 2014, Jul. 2014.

[30] C. Hofner and G. Schmidt, “Path planning and guidance techniques for an autonomous mobile cleaning robot,” in *Proceedings of Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World'*, Munich, Germany, Sep. 1994, pp. 610–617, vol.1.

[31] K. Sato, M. Ishii, and H. Madokoro, “Testing and evaluation of a patrol robot system for hospitals,” *Electronics & Communications in Japan*, vol. 86, no. 12, pp. 14–26, Dec. 2003.

[32] A. Sgorbissa and R. Zaccaria, "Planning and obstacle avoidance in mobile robotics," *Robotics and Autonomous Systems,* vol. 60, no. 4, pp. 628–638, Apr. 2012.

[33] Yootech Technology. (2017, March 31). *Wireless Charger, Yootech Qi Wireless Charging Pad* [Online]. Available: http://www.yootech.net/00019.html.

[34] Adafruit. (2017, March 31). *Universal Qi Wireless Receiver Module* [Online]. Available: https://www.adafruit.com/product/1901.

[35] Arduino. (2017, March 31). *Arduino UNO & Genuino UNO* [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno.

[36] Raspberry Pi Foundation. (2017, March 31). *Raspberry Pi 2 Model B* [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[37] Amazon Inc. (2017, March 31). *Raspberry Pi 5MP Camera Board Module* [Online]. Available: https://www.amazon.com/Raspberry-5MP-Camera-Board-Module/dp/ B00E1 GGE40/ref=sr_1_fkmr3_4?ie=UTF8&qid=1490985717&sr=8-4-fkmr3&keywords= raspberry+pi+camera+1.3.

[38] SparkFun Electronics. (2017, March 31). *Sparkfun Logic Level Converter – Bi-Directional* [Online]. Available: https://www.sparkfun.com/products/12009.

[39] Raspberry Pi Foundation. (2017, March 31). *Help Videos* [Online]. Available: https://www.raspberrypi.org/help/videos/#getting-started-with-raspberry-pi.

[40] Raspberry Pi Foundation. (2017, March 31). *SSH Using Windows* [Online]. Available: https://www.raspberrypi.org/documentation/remote-access/ssh/windows.md.

[41] PyImageSearch. (2017, March 31). *Installing OpenCV 3.0 for both Python 2.7 and Python 3+ on your Raspberry Pi 2* [Online]. Available: http://www.pyimagesearch.com/2015/07/27/installing-opencv-3-0-for-both-python-2-7-and-python-3-on-your-raspberry-pi-2/.

APPENDIX A: SENSING COIL SIMULATION—MATLAB CODE

```
% This function returns the magnetic flux density vector at a point due
to a circular loop of wire with one turn. It uses the Biot-Savart law.
% Inputs are in SI units:
%   I = current in the loop of wire, runs counter clockwise about z
axis
%   R = radius of loop
%   x,y,z = coordinate in space to calculate magnetic flux density
% The origin is at the center of the wire loop

function [ B ] = coilMagField( I, R, x, y, z )
    syms theta;     %Initialize symbolic variable for integration
    r = [x-R*cos(theta) y-R*sin(theta) z];  %Construct location vector
    r_mag = norm(r);    % Location vector magintude
    r_hat = r/r_mag;    % Location unit vector
    dl = R*[-sin(theta) cos(theta) 0];  % Loop wire segment
    f = cross(dl, r_hat)/r_mag^2;   % Integrand of Biot-Savart law
    % Following lines calculate integral in x,y, and z directions
    % If statements are to avoid integrating empty functions
    if(f(1) == 0)
        x = 0;
    else
        x = integral(matlabFunction(f(1)), 0, 2*pi());
    end
    if(f(2) == 0)
        y = 0;
    else
        y = integral(matlabFunction(f(2)), 0, 2*pi());
    end
    if(f(3) == 0)
        z = 0;
    else
        z = integral(matlabFunction(f(3)), 0, 2*pi());
    end
    B = (10^-7)*I*[x y z];  % Multiply by constants of integration
end
```

```
%% This script finds the zero crossing of Bz (the z component
% of the magnetic flux density) due to a circular loop of wire.
% This calculation is done at different heights.
% The zero crossing is found using a bisection method with the
% tolerance specified in the code.

% This script makes use of the coilMagField() function

close; clear; clc;
% Select coil parameters
R = 1;  % Radius of wire loop in meters
I = 1;  % Current running through loop of wire
```

```
samples = 20;   % Number of heights to calculate and plot
delta = R/(samples-1);  % Space between heights in meters

% The following lines find the zero crossing at each height
for(i = 1:1:samples)
    h = (i-1)*delta;
    lb = 0;     % Initial left bound for bisection method
    rb = 4*R;   % Initial right bound for bisection method
    dx = 1;
    x0 = 0;
    % Bisection method with stopping critera
    while(abs(dx)>0.001)
        x = (lb+rb)/2;
        B = coilMagField(I, R, x, 0, h);
        if(B(3)<0)
           rb = x;
        elseif(B(3)>0)
           lb = x;
        end
        dx = x-x0;
        x0 = x;
    end
    Bz(i) = B(3);
    H(i) = h;
    Z(i) = x;
end

plot(Z,H,'k-o');
ylabel('Height from primary coil in primary coil radii');
xlabel('Zero-crossing location in primary coil radii');
xlim([0 1.8])
grid on;

-----------------------------------------------------------------------
%% This script calculates and plots the induced voltage in a coil
% parallel to a circular loop of wire. This calculation is performed
% for several misalignment distances.
% The height and diameter of the secondary coil are specified below and
kept constant.

% This script makes use of the coilMagField() function

close; clear; clc;

% Select primary coil parameters
R = 1;  % Radius of primary coil in meters
I = 1;  % Current running through primary coil

% Select secondary coil parameters
d = 3*R/4;  % Secondary coil diameter in terms of primary coil radius
h = 8/8;    % Height of the secondary
```

```
% Select sampling resolution for flux calculation
n = 10;          % Even number, higher number yields higher resolution
delta = d/n;     % Space between sampled points
points = 60;     % Number of dislocation values to plot

% Sample and plot calculated fluxes at different dislocations
for(k = 1:1:points+1)
    cx = (k-1)*3*R/(points);
    Bz_sum = 0;
    for(i = 1:1:n)
       x = cx-d/2+delta+(i-1)*delta;
       y = delta/2;
       rightBound = sqrt((d/2)^2-(cx-x)^2);
       while y < rightBound
            B = coilMagField(I, R, x, y, h);
            Bz_sum = Bz_sum+B(3);
            y = y + delta;
       end
    end
    X(k) = cx;
    flux(k) = abs(Bz_sum*2*delta*delta);
end

plot(X, flux, '-o')
grid on
xlabel('Distance away from primary coil center in primary coil radii')
ylabel('Flux through sensing coil [Wb]')
```