Achieving Stable Magnetic Levitation On The Arduino Platform

Michael O. Morken and Brett R. Haffner

Department of Physics, Wheaton College, Wheaton, IL 60187

(Dated: February 5, 2015)

Abstract

We use a Proportional, Integral, Derivative (PID) controller implemented on the arduino microcontroller to control the current through an electromagnet based on the position and velocity of the magnetic object that is being levitated. The electromagnet was connected to a 12V DC power supply and it’s current was controlled through a MOSFET transistor connected to a PWM pin on the arduino. The PID controller would determine the duty cycle of the PWM pin and thus would vary the current based on the position and velocity of the levitating object. The position and velocity of the levitating object were quantified optically by using an infrared LED and a phototransistor that was sensitive to infrared light. The position of the object would correlate to a certain brightness of the phototransistor and the velocity would be calculated by finding the change in position since the last time through the control loop.
I. INTRODUCTION

Anyone who has ever played with magnets has experienced the repulsive force when the north pole of one magnet is brought into close proximity with the north pole of another magnet. One may have wondered if it is possible to configure the magnets in such a way as to have them levitate one above the other. You know if you have ever tried this, that the magnets will flip around and stick to each other north pole to south pole. In fact it is impossible to get a system of permanent magnets to achieve stable levitation. This fact was first proved by Samuel Earnshaw in 1842, and is now known as Earnshaw’s theorem. So while magnetic levitation with only permanent magnets is impossible it is still possible to levitate a permanent magnet. There are four known loopholes to earnshaw’s theorem. Thee loopholes are not instances where earnshaw’s theorem is disproved but rather they involve phenomena observed in materials that cannot be classified as permanent ferromagnets. The four loopholes are (1) Gyro-Fixation, (2) diamagnetism, (3) superconductors exhibiting the meissner effect, and (4) current modulated electromagnet. In this paper we will be describing the process of achieving stable magnetic levitation through the use of a current modulated electromagnet.

II. THEORY

Setting up a current modulated electromagnet is not as simple as just connecting a coil to a power supply and placing a permanent magnet into the resulting magnetic field. If the current in the electromagnet isn’t constantly changing then the magnet will just flip over as if the coil was a permanent magnet. The key to the stable levitation is the current modulation. The current through the coil of needs to change as the position of the magnet changes. As the magnet is attracted to the coil the current in the coil needs to decrease and as the magnet falls further from the coil the current in the coil needs to increase. This process needs to be happening fast enough so that the magnet appears to be held in one place even though in reality it is moving slightly towards and away from the coil while the coil current is constantly adjusting itself to keep the magnet as close to a set position as possible.

In order to successfully levitate a permanent ferromagnet by means of a current modulated
electromagnet there are a few systems that will have to seamlessly work together. The first system is a method of detecting the position of the magnet. In order to be able to adjust the coil current based on the position of the magnet we must be able to measure the position of the magnet very accurately and quickly. Once the position is determined we will need an algorithm to determine how the current must be changed in order to move the magnet back to it’s desired position. Finally we will need a system that is capable of quickly changing the current in the electromagnet based on the result of the algorithm. All these systems must be able to work together very quickly in order to levitate the permanent magnet.

III. HARDWARE OVERVIEW

The device that we will use as the foundation for all of the systems that we described above is the arduino UNO microcontroller. This device is popular choice and is essentially a breakout board for the ATmega328 chip. The board has the capability to digitize analog inputs with a 10 bit ADC and a sampling rate of $10kHz$. The board also has the capability of outputting an analog signal through it’s digital output pins by a process termed Pulse Width Modulation.

FIG. 1. The analog signal is represented by discrete pulses of a digital output pin.
These characteristics will allow the position of the magnet to be quantified by using the arduino’s analog to digital converter and then the signal can be sent out to the coil through one of the arduino’s PWM pins. The arduino also will be able to execute a program where we will be able to implement an algorithm that will be able to hold the magnet in place and adjust and correct for tiny oscillations in the magnets position.

IV. HARDWARE SETUP

Using the arduino we were able to construct all of the systems that were necessary to achieve magnetic levitation. These systems involved interfacing with hardware sensors, making calculations, and outputting signals to change the state of the system.

FIG. 2. A Picture of the hardware setup.

A. Measuring The Position Of The Magnet

In order to determine the position of the magnet we opted to use an infrared light sensor in combination with an infrared LED. The infrared light sensor will be a phototransistor.
This will provide an analog input that we will be able to digitize with the Arduino’s ADC. The setup of the LED will have to be directly in line with the phototransistor. This will make the value of the phototransistor represent the position of the levitating magnet.

FIG. 3. hardware setup for measuring the position of the levitating magnet.

As the magnet rises up towards the coil (not pictured in the figure) it blocks more and more of the light that the LED is sending out in all directions. Thus the phototransistor
registers less brightness. With some experimentation it was shown that the brightness of the phototransistor corresponds to the position of the magnet. One thing that had to be done on the software side of this setup was that the signal of the phototransistor needed to be inverted to give desirable measurements. This phototransistor registers a higher voltage when more light was blocked. So as the magnet traveled upwards the signal was increasing. This would have correlated to more power to the coil which would have worsened the situation by introducing an unwanted positive feedback loop. So before the digitized value of the analog input was passed to the control algorithm it was first inverted. This was done with the following line of code:

```c
int raw = 1023 - analogRead(sensorPin);
```

This was effective because the sensor would read 1023 when it was completely covered and zero when it was receiving all of the light. So $1023 - 1023 = 0$ which is a very small signal when all of the light is blocked and a very large signal (1023) when the light is hitting the sensor. This allows the position to be recorded by the arduino correctly so that it can be used in the control algorithm that will perform calculations to keep the magnet levitating steadily.

Another aspect of the position measurement that we implemented was a method of compensating for changes in the ambient light. In order to do this we had to measure the ambient light (the value of the phototransistor when the LED was off) and subtract that from the measured value. However this can’t be done too often so it won’t interfere with the control loop that keeps the magnet at a fixed position. We opted to record the ambient light every thousand times through the control loop. So the value of ambient light that is measured will be used for the next thousand times through the loop before it will be re-measured. The function that we used to record the ambient light is shown below:

```c
int readAmbient()
{

digitalWrite(irPin, LOW);
    // allow time for LED and phototransistor to settle
delayMicroseconds(100);
    int ambient = 1024 - analogRead(sensorPin);
}```
Here it can be seen that the LED is switched off and the phototransistor is read after things settle down. This comprises the system that determines the position of the magnet.

B. PID Control Algorithm

The control system that we used in order to adjust current flowing to the electromagnet was a Proportional Integral Derivative (PID) controller. The PID control system uses a setpoint, a desired state of the system, and then calculates error by comparing the systems current state against the setpoint to make appropriate changes in order to move the system towards the setpoint. The algorithm used for the PID controller is:

\[
\mu(t) = K_p e(t) + K_i \int_0^t e(\tau) \, d\tau + K_d \frac{d}{dt} e(t),
\]

where \(\mu(t)\) is the output to the controller, \(e(t)\) is the error as a function of time, and \(K_p, K_i,\) and \(K_d\) are the coefficients for the proportional, integral, and derivative components of the controller. The value of these coefficients is where the magic of the PID controller lies. The values of \(K_p, K_i,\) and \(K_d\) are manually adjusted until the system responds quickly enough and is up to the required standards. When we were tuning the PID controller it was essential to understand the effect that each of these terms has on the system.

The proportional term determines the magnitude of response to the error. For a large value of \(K_p\), the controller will output a large signal with small error present. If \(K_p\) is small, the controller will not respond to large error. Essentially this is just a multiple of the actual value of the error. In our case term needs to be large enough that the system will give the electromagnet enough current to levitate the magnet but not so much current that it sucks it up to the coil without having enough time to respond. The integral term adds up the error over time, through the integral, and multiplies the accumulated error by \(K_i\) to correct any error that hadn’t been corrected in the past. Essentially this term compensates for the past positions of the magnet and will adjust accordingly as the integral changes. If the magnet was oscillating around the setpoint this term would go to zero. It mainly
deals with removing large initial errors. Finally, the derivative term calculates the change in the error, depending on the value $K_d$, the controller will output an appropriate signal based on the change in error. This essentially allows the system to know the velocity of the magnet. So the derivative term will allow us to hold the magnet at the setpoint with zero velocity. This term will get rid of unwanted oscillations that could have been introduced by the proportional term.

In our case, our desired setpoint is the position where we wanted the levitating magnet to remain stationary. The setpoint was determined to be 25 by reading the sensor pin on the arduino to determine the objects desired position. This value was determined experimentally reading data on the arduino through the serial monitor. The specific code to implement this was, `int position = analogRead(sensorPin);`. The error term was determined by subtracting the position from the error:

```c
int error = position - setPoint;
```

The derivative error was calculated through subtracting the new error from the old error, which was the error that was calculated on the previous run through the control loop:

```c
int derivative = error - oldError;
```

Finally, the integral term was calculated by summing the error from every time through the control loop:

```c
int errorSum = 0;
errorSum += error; // Within the void loop
```

All three of these were combined, to determine the power output to our electromagnet through:

```c
int power = Kp*error + Kd*velocity + floor(Ki*errorSum) + constantPwr;
```

with the addition of the term `constantPwr` to continue to supply current to the electromagnet in the rare case that there was no error present in the control system. The constant power is necessary because with this system even if the magnet is perfectly at the setpoint it does take some power in the coil to hold it there. The coefficients $K_p, K_i$, and $K_d$, were determined by
experimentation. We tuned the PID controller in order for the magnet to be stable when levitating. The values used in our code were:

\[ K_p = 2, K_d = 90, K_i = 0.1. \]  \hspace{1cm} (2)

C. The Electromagnet

The PID controller calculated the appropriate output signal to hold the magnet in place but we needed a way to get this signal to the electromagnet. The arduino can only output 5V and a measly 20mA of current. This is not enough to control the electromagnet. To overcome this problem we opted to use a MOSFET transistor that was capable of controlling a larger 12V and 2A power supply that would drive the electromagnet. The mosfet allowed the arduino’s digital output pin to control the much larger power source.

![n-Channel MOSFET](image)

FIG. 4. The MOSFET that allowed the arduino’s output pins to control a much larger power supply.

The arduino interfaces with the electromagnet through a MOSFET transistor. The arduino’s digital output pin is connected to the gate and the power flows through the source to the drain. We used an n-Channel MOSFET in our circuit. The use of a transistor allows the
small voltage of the arduino microcontroller to control a much larger power supply. This transistor allows the pulse width modulation of the arduino, which ranges from 0 – 5V to source current from the 12V DC power supply. The coil that made up the electromagnet was hand wound with 20 gauge insulated copper magnet wire. The core of the electromagnet was a steel threaded rod so it is a ferrite core electromagnet. The resistance of the coil was 4.1Ω.

FIG. 5. The electromagnet was a hand wrapped coil and measured 4.1Ω.
D. Bringing it All Together

To hold the various components of this project together we needed a frame to support the electromagnet, the phototransistor and the photodiode. To accomplish this task we designed a frame that we 3D printed out in PLA, a type of plastic. This frame maintained a distance of 90\text{mm} between the infrared LED and the phototransistor and allowed the position of the electromagnet to be adjusted easily in the tuning process. The advantage of 3D printing this component is that the frame was designed to fit the electrical components perfectly.

FIG. 6. The full setup in action.
V. UNFORESEEN DIFFICULTIES WE OVERCAME

A. PID Tuning

The most time consuming part of this project was the tuning of the PID controller. Our process for tuning the PID controller, while not perfect, yielded the desired result. We first worked with the constant that was added to the PID algorithm to keep the magnet levitating steadily. This was done by holding the magnet and feeling the point at which the magnetic force from the coil offset the weight of the magnet. Once the magnet was somewhat balanced the proportional term was adjusted until the magnet would levitate for a short time on its own while oscillating heavily. Once the proportional term was set the derivative term was increased until the oscillations of the magnet were removed for the most part. The integral term was added to remove error in the initial position of the setpoint and was adjusted while initially holding the magnet at different positions. This process allowed the PID controller to be tuned to within a functional threshold.

B. Noisy Power Supply

A major problem that we overcame was the presence of 60Hz noise in our Triad wall wart power supply. This power supply was powering the electromagnet and it introduced oscillations into the system that were not based on the PID algorithm. In an attempt to correct this we swapped the power supply with a bench top supply that supplied constant voltage at 12V.

The new power supply fixed our problems with unwanted oscillations and allowed the system to achieve the stability that was desired. Further stability could be achieved through the use of a low pass filter designed to remove 60Hz noise that was still persisted in the system.

C. PID Frequency

Another problem that we ran into was how often to call the PID algorithm. Originally our thought was to call the algorithm as fast as possible. However we quickly realized that this would be problematic. Every time the PID algorithm was called it would recalculate
the output to the PWM pin and it would change the duty cycle of the PWM pin. This was a problem because each time the duty cycle of the PWM pin was changed it would start with a high pulse of $5V$. When this was done rapidly it essentially created a consistent $5V$ signal that overwrote the PID calculations. To compensate for this effect we put a delay into the code so that the PWM duty cycle would have some time to run before it would be overwritten for the next calculation. We ended up delaying the loop for 700 microseconds.
before refreshing the PWM pin’s duty cycle.

VI. CONCLUSION

After successfully levitating a magnet, we conclude that the PID controller is an effective way to minimize error for a system in a desired state. The Arduino hardware platform is also able to implement this algorithm fast enough to keep a magnet in stable levitation. Future applications of this work would be to create multiple setpoints so that the magnet could move up and down for visual effect. Using our method of quantifying position, this would require multiple LEDs and phototransistors that would run the length of the stand. Another thought would be to add a coil at the bottom opposing the top coil to create more stability. This could also potentially allow for a greater range of stable motion for the object being levitated; however, it would make the PID algorithm much more complex. In order to create a more practical device that could do things like levitate objects placed on a table a group of 4 coils arranged in a magnetic quadrupole could be used to hold a magnet in place.

FIG. 9. Example of a magnetic quadrupole created with coils. This type of configuration could be used to create a similar device that levitates on object above the coils rather than “hanging” below a single coil.

The PID algorithm also has wide ranging applications to controlling all kinds of systems.
that are maintained at a fixed setpoint. So our method of determining the PID constants could be used in a variety of disciplines.