EE 90 Final Report

Ajay Mandlekar
California Institute of Technology

December 8, 2015

1 Motivation

The purpose of this project is to make a dog bowl that has the capability to control access to it based on the physical proximity of the dog to the bowl and the identity of the dog.

This dog bowl would be very useful in households with multiple dogs where different dogs need different food, or if different dogs should be fed at different times. It is also useful for making sure that only dogs are allowed access to the food inside, protecting the food from insects, cats, and children.

2 Overview

The main technology used in this project is infrared (IR) LEDs and photodiodes. Each dog tag (transmitter) transmits a square wave at 3.56 kHz using an IR LED. The dog bowl has several photodiodes to receive this signal. This signal is then amplified and then passed through a filter. The filter cuts off frequencies that are significantly different from 3.56 kHz - this is how identification is achieved. A dog tag that has a much lower or higher frequency (356 Hz or 35.6 kHz, for example) would correspond to a dog that does not have access to the bowl.

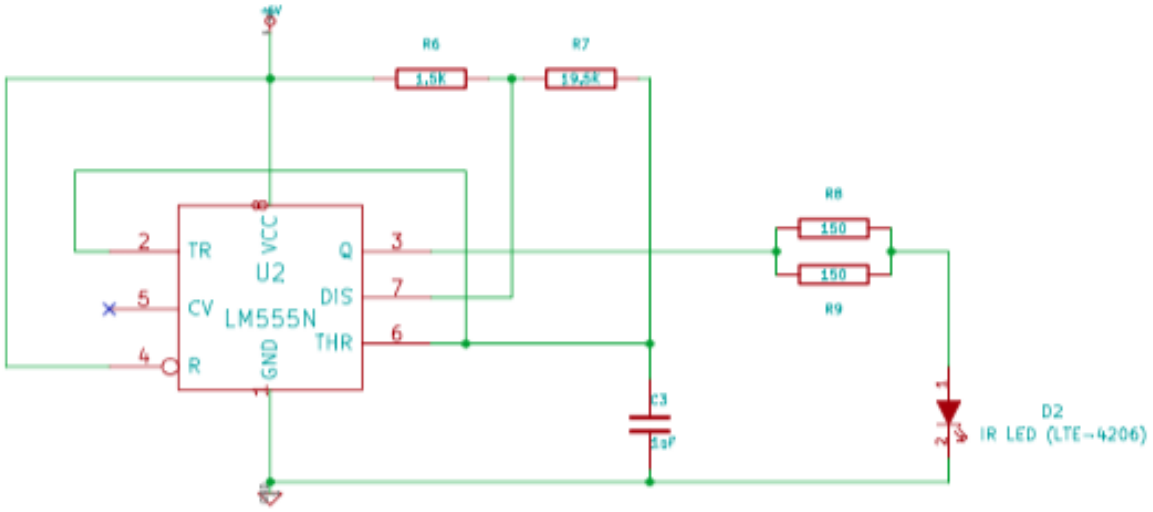
After the filter stage, the signal is passed through a second amplifier and a half-wave rectifier. It is then passed to a microcontroller (μC) on its analog input input. The microcontroller uses its analog-to-digital converter to measure the received voltage. If the voltage is high enough, it sends a control signal to a servo motor in order to open the dog bowl. It should be noted that the microcontroller runs some additional logic in software in order to close the bowl when a signal is not detected.

3 Specifications

1. The dog bowl opens up when a dog tag is about 6 inches to 1 foot away from the bowl, at roughly the same vertical height. The dog tag needs to be transmitting within the correct range of frequencies.
2. The transmitter has an LED that needs to be pointed towards the front half of the dog bowl, similar to how a TV remote needs to be pointed towards the TV.
3. Both the dog tag and the dog bowl are battery powered. The dog tag is powered by two 3V coin cells, and the dog bowl is powered by two 3V coin cells and a 9V battery.

4 System Components

4.1 Transmitter



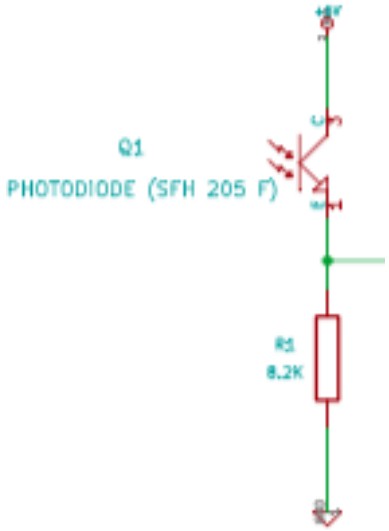
The dog tag simply uses an LM555 timer to generate a square wave. The frequency of the square wave is given by

$$\frac{1.44}{(R_A + 2R_B)C}$$

Here, we have used $R_A = 1.5k\Omega$, $R_B = 19.5k\Omega$, and $C = 0.01\mu$F in order to generate a frequency of 3.56 kHz. The output of the timer is connected to a 75Ω resistor and the IR LED. The value of 75Ω was chosen to bias the IR LED to transmit as much power as possible. This was done by considering the maximum power of the IR LED, 90 mW, and its forward voltage ranges, 1.2-1.6 V, and choosing the smallest resistor value that results in power under the maximum power limit.

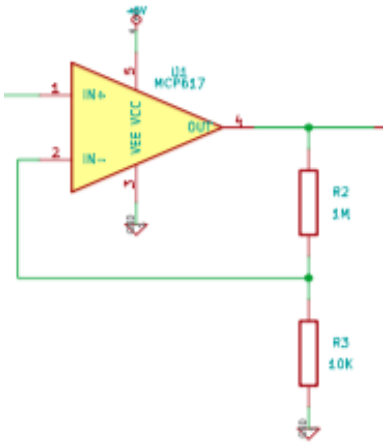
4.2 Receiver

4.2.1 IR Sensing



To receive the signal from the dog tag, the dog bowl has 4 photodiodes around its perimeter. These photodiodes generate current when there is incident IR light. This current is turned into a voltage by the $8.2k\Omega$ pull-down resistor. This is how the square wave is received by the dog bowl.

4.2.2 First Gain Stage

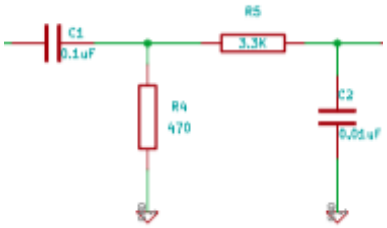


This is a simple non-inverting op-amp gain stage. The op-amp's high input impedance made this circuit a very convenient choice as a gain block. The gain of this circuit is given by

$$\frac{V_{out}}{V_{in}} = 1 + \frac{R_B}{R_A}$$

Here, $R_B = 1M\Omega$ and $R_A = 10k\Omega$, to produce a gain of 101.

4.2.3 Bandpass Filter



This is simply an RC highpass filter and an RC lowpass filter cascaded together. The transfer function of this arrangement is given by

$$H(s) = \frac{R_1 C_1 s}{1 + R_1 C_1 s + \frac{R_1}{R_2} + R_1 R_2 C_1 C_2 s^2 + R_2 C_2 s + C_2 R_1 C_1 s}$$

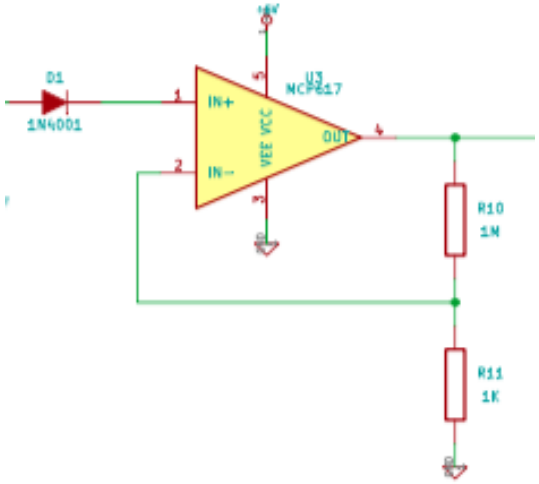
where R_1 and C_1 correspond to the highpass filter (the left half of the diagram) and R_2 and C_2 correspond to the lowpass filter (the right half of the diagram). Choosing $R_1 \ll R_2$ and C_1, $C_2 \ll 1$ allows us to factor the transfer function as

$$H(s) = \frac{R_1 C_1 s}{(1 + R_1 C_1 s)(1 + R_2 C_2 s)}$$

which corresponds to the transfer function of a bandpass filter. Therefore, we must have $R_1 \ll R_2$ and C_1, $C_2 \ll 1$ in order to make sure that cascading the lowpass filter after the highpass filter is okay.

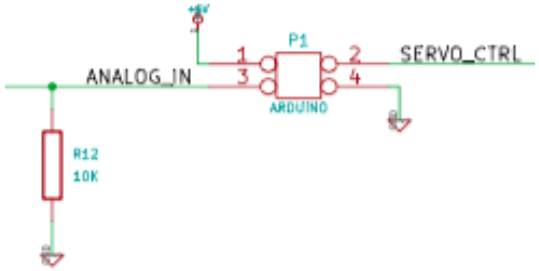
The lower cutoff frequency is given by $f_L = \frac{1}{2\pi R_1 C_1}$ and the upper cutoff frequency is given by $f_H = \frac{1}{2\pi R_2 C_2}$. In order to satisfy the constraints while choosing reasonable cutoff frequencies, the following values were chosen - $R_1 = 470\Omega$, $R_2 = 3.3k\Omega$, $C_1 = 0.1\mu F$, $C_2 = 0.01\mu F$. This resulted in a lower cutoff frequency of about 3.4 kHz and an upper cutoff frequency of 4.8 kHz.

4.2.4 Second Gain Stage



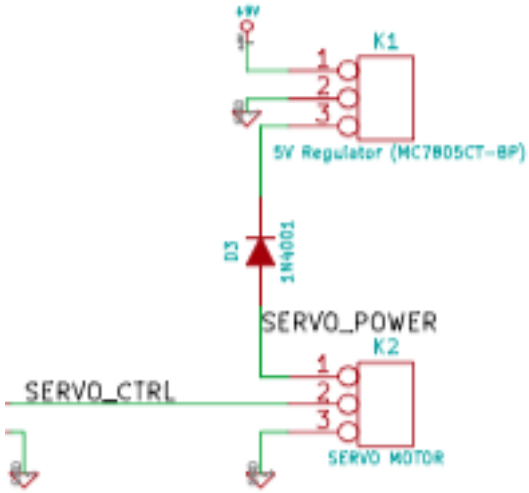
The second gain stage is nearly identical to the first. The main difference is that it receives a pure sinusoid from the bandpass filter. However, the op-amps are single supply. In order to fix this problem, a half-wave rectifier is used to block the negative portion of the sinusoid. The rectified signal is then amplified by the non-inverting gain block, which is configured to have a gain of 1000.

4.2.5 Microcontroller



An Arduino Uno is used in order to interface with the servo motor. A pull-down resistor is used on the analog input line in order to make sure that it was never floating. The Arduino essentially keeps sampling voltages on its analog input line using its ADC. Once the voltage is above a threshold (set to about 0.5 V) it outputs a PWM (pulse width modulated) signal to the servo motor to open the bowl. The Arduino keeps the bowl open until a timeout occurs, which happens 5 seconds after the Arduino stops receiving a voltage past its threshold. At this time, the Arduino outputs a PWM signal to the servo motor to close the bowl.

4.2.6 Servo Motor



The servo motor is responsible for physically opening and closing the dog bowl. It is powered by a 9V battery that is regulated down to 5V. The servo motor uses a separate power supply due to the large amounts of current that it draws. It was also necessary to place a reverse-biased diode between the power lines of the servo motor to prevent noise from appearing on other parts of the circuit.

5 Problems

I faced many significant problems during this project.

1. I had some silly problems with my simple bandpass filter. I first tried cascading a highpass and lowpass filter together with little to no thought on interacting poles. After that didn't work, I actually did the transfer function analysis and came up with conditions to ensure that cascading the filters would work. I tried again with appropriate resistor and capacitor values and it still didn't work. Then, I realized that I forgot to divide by 2π to convert from angular frequency to frequency. It finally worked after that.

2. My servo motor was broken out of the box. I tried my friend's servo with my test program and it worked but my servo had a mechanical problem that I could not fix despite opening it up and trying to find out what was wrong with it. Luckily my friend had a spare.

3. Connecting the servo motor to my receiver circuitry was introducing a lot of noise, and made IR detection impossible. I first tried giving the servo motor its own supply, but that wasn't enough to fix the issue. Then, I tried to fix the problem in software by having the Arduino disconnect the servo every single time it wanted to read the voltage. This was inconvenient and made refreshing the timeout impossible. Finally, I placed a reverse-biased diode on the power supply of the motor to drain away current that was probably generated by the back emf from the motor. This fixed the noise issue.

4. While testing my receiver, I accidentally moved my Arduino a little too close to my 9V battery and a short was created. Luckily, my circuitry was not harmed but the 9V battery was burning hot for about 10 minutes.

6 What I Learned

During the course of this project, I learned many things. I had never before created a finished product, from beginning to end. The closest I had come to doing something like this was EE 52, but the priority there

was getting it to work and there was a significant emphasis placed on documentation. This was the first time that I had to think about what would make my project look nice and polished - it is a very useful skill to have and it is often neglected in engineering classes. In the real world, it isn't enough for a product to work. It needs to have aesthetic appeal - this is why Apple is so successful. Other than learning about the importance of packaging, I learned many other things in this class.

1. How to use batteries in electronics projects to make them portable.
2. How to solder surface mount battery holders onto a protoboard (with a thin piece of wire and lots of patience).
3. Avoid using coin cells if possible. They run out of juice very quickly and are hard to take out of their holders. Using 9V batteries with regulators worked very well on the servo supply, and I would definitely use them on the other supplies if I had more time.
4. It would have been very useful to have LEDs on my transmitter and receiver to indicate that they were on so that I could avoid leaving them on. As I mentioned before, the coin cells drained quickly and are nearly impossible to remove without destroying their fragile connections.
5. Next time, hot glue the power switches, especially the one on the transmitter. It is a little bit lose, but I didn't want to undo the packaging before doing the demo.
6. Don't use strong single gauge wire if possible. I thought it would be more convenient to solder, but the wires kept breaking very often. Stick with normal, flexible wire that has several strands. This is ultimately how I was able to route my photodiodes to the dog bowl from the receiver circuitry.
7. Single supply op-amps make for awesome gain stages without the hassle of dual power supplies and worrying about biasing BJTs correctly. However, the application needs to be relatively low frequency and single supply op-amps don't come cheap.
8. It is sometimes possible to solve, or circumvent, hardware problems in software, like using Arduino software to circumvent the servo motor noise problem. However, it is ultimately better to find an actual solution to the problem rather than a quick workaround.
9. Theory is great, but in practice, you often need to try things instead of thinking too hard. Experimenting with different values for different components was very useful. This also helped when choosing values for different variables like the timeout and threshold values on the Arduino microcontroller.
10. It is always a nice idea to keep a notebook. It helped me remember what I did and when, what values I used, and what I need to do next time. It also helped me write this report!