Chook Gate

Team B4

The Automated Australian Chook Gate

Date: May 4, 2009

Authors:
Lilian de Greef
Mark Ellis
Nick Hill
Nicole Peck

---

1. Abstract

Rodney Shannon of Rodney Downs Pastoral Company assigned the task of designing a device that would automatically open and close the gate of his chicken coop to a group of four Harvey Mudd College engineering students. They worked through the design process in three steps. First, they explored the design space by defining and creating metrics for the design's functions, objectives, and constraints. The design team split the design into three functionally independent subsystems, which were then optimized and synthesized. The resultant design involved a leadscrew that determined the position of a sliding door with the use of a motor, all of which are controlled by a microcontroller and various sensing devices attached to it.

TABLE *of* CONTENTS

LIST *of* TABLES

TABLE *of* FIGURES

Technical Memorandum

To: Rodney Shannon
From: Lilian de Greef, Mark Ellis, Nick Hill and Nicole Peck
Subject: Design of Automated Chicken Coop Gate

2. Introduction

The design team was given the task of designing a device to open and close a chicken gate automatically each day. The purpose of this device is to minimize the client's necessary involvement while still allowing the chickens freedom outside of their coop for an adjustable amount of time. This technical memorandum documents the process used by the design team that led to a final design and prototype.

The process involved three main steps: understanding the design space, devising designs, and creating a final design. To understand the design space, the team researched relevant background information and defined functions, objectives, and constraints to help them generate possible solutions. To devise design alternatives, the team separated the design space into three subsystems: gate-opening, timing, and powering. They then proceeded to generate alternatives for each subsystem and combine the best of each into one final design. From this, the team built a prototype of their chosen design to demonstrate its feasibility and tested it to discover any possible improvements for the client to consider.

3. UNDERSTANDING THE DESIGN SPACE

Before brainstorming designs, the team took some time to formally understand every aspect of the challenge. To do so, they determined exactly what tasks the final solution should perform, its desirable characteristics, and the requirements it must meet. In other words, the design team defined the device's functions, objectives, and constraints to help formulate design alternatives.

In order to determine these aspects of the design, the team first had to gain more insight into the problem through communications with the client, research regarding the environment a gate-opener would be operating in, and other general information about the design space as a whole.

3.1 Problem Statement

The design team was given an initial problem statement from the client, Rod Shannon, describing the problem he wanted solved. This entailed relevant details to the problem, his current gate's specifications, and other background information necessary to understand the problem and the reason it warranted a solution.

The client's current gate is an approximately six foot by four foot construction made of steel frame covered with chicken wire as Fig.1 illustrates. It is a manually-operated gate that requires a person to open and close it every day to let the chickens in and out of the coop and is therefore somewhat of a hassle especially if no one is consistently present every morning and night.

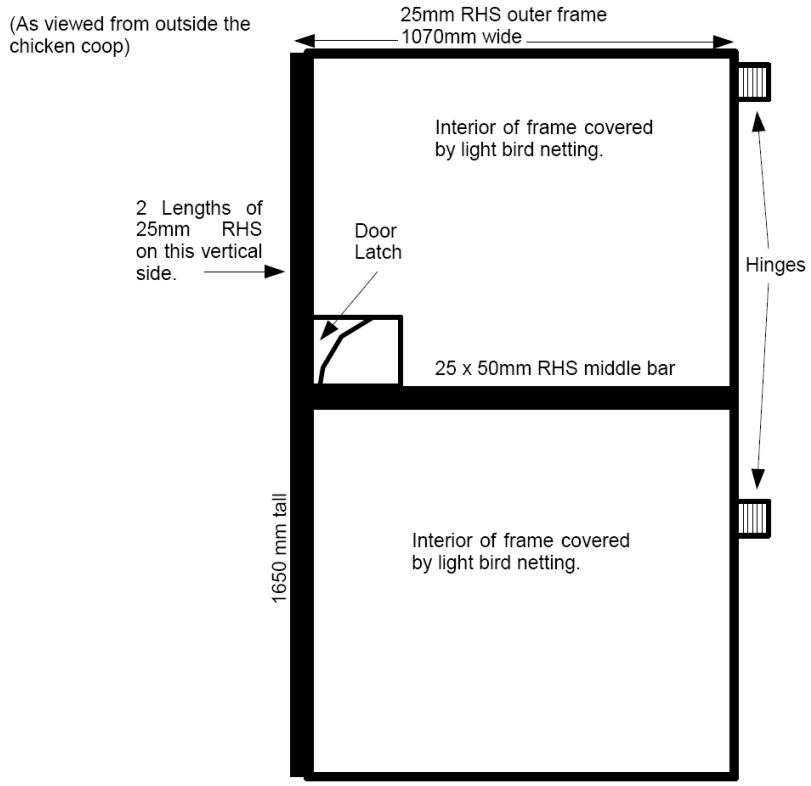


Figure 1: Specifications of client's current chook gate.

Although possible solutions include keeping the chooks confined within an enclosure at all times, building a large structure for them to roam around in (removing the need to let them out), or even letting them run wild with no cage at all, there are several constraints that disqualify these options. One, chooks require a lot of exercise; keeping them enclosed at all times is not healthy or humane. Two, a permanent structure would be quite massive and would quickly exhaust the area covered by the enclosure. Three, foxes are a serious threat to chickens in Australia. They have a very keen sense of smell and they hunt at night when chickens would be roosting, so it is not an option to let the chickens roam free; they would be easy prey to foxes without some sort of protective enclosure [8].

Given these constraints, one can see the need for some sort of solution to the problem presented. The chooks need to exercise and be let out of their coop but they can only be out for a couple of hours before they ruin the garden, and they need an enclosure to protect them from predators such as foxes. However, the chickens' needs must be balanced with the client's time constraints and availability to watch over them and let them in and out of the coop.

For all of these reasons, the design team was presented with the task of designing an *automated* gate-opening device that would open and close by itself at user-designated times. This would afford the client the best of both worlds: he can keep the current chook coop and also specify how long he wants the chickens out. Additionally, since the gate would be automated, it would require very little involvement and could be left alone to open and close consistently with little to no supervision. After consultation with the client and more research into the problem, the design team took all of the information and synthesized it into what is known as the problem statement for the design. A design's problem statement is a concise and pointed statement that addresses the core of the task at hand. Its purpose is to define the design space for the rest of the design process—to give the team direction in how to proceed, and reads as follows:

> *Design a gate-opening device for a chook-yard that lets the chickens out at an adjustable time, remains open, and closes an adjustable time after sunset each day. The gate must remain closed at night to protect the chickens from predators such as foxes. The device must not prevent human access and should not be powered by the electrical grid (preferably solar-powered).*

3.2 Background Information

The team had a direction to travel in but still needed more information relevant to the problem. For example, since a final design would be implemented on the client's ranch in Ilfracombe, Australia, the team needed to know what kind of environmental conditions it would be dealing with there. In addition, since sunlight would be used to power the final design and since the gate's timing could be based off of sunrise and/or sunset, the team needed to know a lot more about the sun and its time variance throughout the year.

The team researched what kind of environmental conditions a design would deal with on a day-to-day basis if it were to be implemented on the client's ranch. Using data from the nearest weather station 16.5 miles away in Longreach, Australia, the team found that the maximum amount of precipitation the city receives in its wettest month is about 80 millimeters (~3 inches), temperatures range from 40° F to 100° F in a given year, and given that the location of Ilfracombe is in the Australian Outback, dust and wind are expected environmental factors [2][3]. Since a final design will most certainly have some amount of moving parts (the door at the very least), and quite possibly several electrical components, precipitation, temperature, wind, and dust are all factors that should be given weight when brainstorming and choosing among design alternatives.

The design team found that sunrise, sunset, and hours of light vary widely over the course of the year. As Fig. 2 shows, in summer, sunrise is earlier and sunset is later resulting in about two more hours of daylight than at the peak of winter.

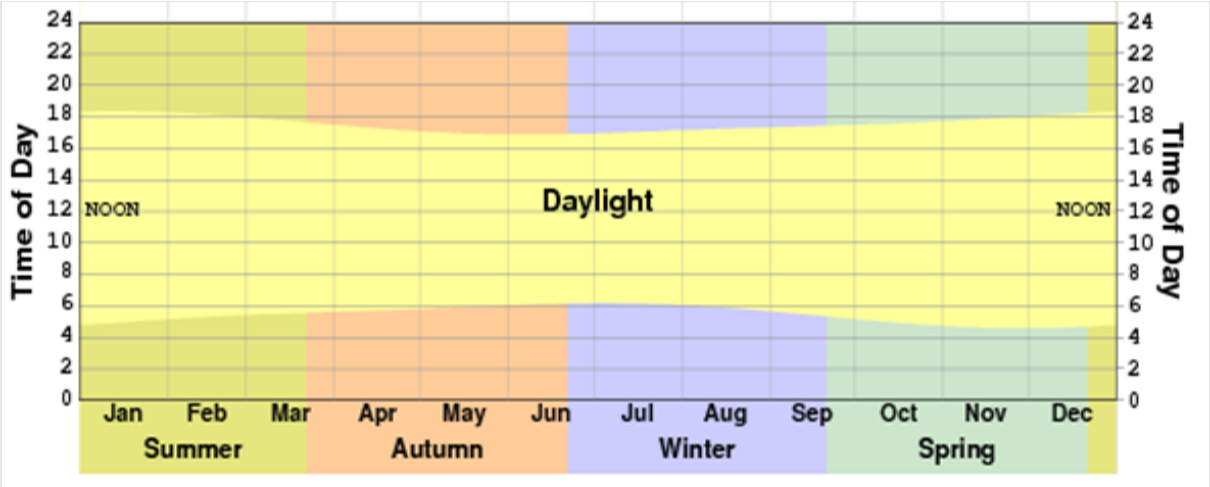


Figure 2: Daylight over the course of a year in Longreach, Australia.
Source: http://www.climate-charts.com/Locations/a/AU94000000360300.php

This seemed to suggest that if a sun-based timing mechanism was used, it would have to be based off of one reference point or constantly adjusted to account for the variance in sunrise and sunset.

3.3 Functions

As mentioned above, the design team developed and organized a set of functions of the device. A function is an "action for which a person or thing is specially fitted or used or for which a thing exists; one of a group of related actions contributing to a larger action," [1]. Less formally, it is what a device or system is meant to do. Explicitly defining the functions of the device simplified the design process by giving the design team a concise and complete reference of what each design needs to accomplish.

To assist in the process of defining functions of the device, the design team agreed that the device must undergo the following cyclical process: at a time during the day that is set by the user, the device must let the chickens out of the coop. The gate must close at a time around sunset that is also set by the user, and must keep the chickens safe from predators until daylight. At this point, the cycle repeats. After defining this process, the design team was able to add more detail to the functions the gate must perform. For example, the design team found that there are three steps to opening a chicken coop gate. The device must 'unlatch' and release the gate from its closed position, move the gate open, and stop it at the open position. The need to close the gate at approximately sunset suggested that the device should be able to detect sunset, and the user's specification that the gate should open and close at adjustable times showed the team that the design needed some sort of time-keeping mechanism with a user interface so that the opening and closing times could be adjusted. Thinking about the daily process a chook gate would go through was a very helpful tool that revealed additional functions the final device needed to perform.

After expanding the list of functions by brainstorming and paring down that list through discussion, the design team organized the functions into a functions/means tree which is shown in Appendix 1. Table 1, below, gives a detailed list of the functions that the design team decided upon.

Function	Description
Collect Energy	The device must gather energy to use in its operation.
Store Energy	The device must store the energy it gathers.
Keep track of time	The device must know when to let chickens out of the coop and when to lock them inside of the coop.
Accept user input	The device must have an interface which allows the user to adjust its operation.
Store user input	The device must be able to store the preferences of the user to adjust its operation.
Detect sunrise/sunset	The device must have an interface which allows the user to adjust its operation.
Process input from sensors	The device must synthesize the information it receives from sensors in order to operate properly.
Connect timing system to actuating system	The theoretical timing of the gate opening and closing must be linked to the physical gate opening and closing.
Release gate from closed position	The device must be capable of transitioning from a state of protecting the chickens from other animals to a state in which animals may enter and exit through the gate.
Move gate to/from open position	The device must actuate and from a position such that chickens may pass in and out of the chicken coop through the gate.
Resist closing forces	The device must remain in the open position.
Stop gate at closed position (lock)/Resist opening forces	The device must be capable of protecting the chickens from foxes by keeping the foxes out of the chicken coop.

Table 1: Functions that the design team specified that the device must perform, as well as a description of each function.

3.4 Objectives and Constraints

To further understand the design challenge, the team defined a set of objectives and constraints for the device. Objectives are desirable but unessential characteristics of the design that are desired but not required. In contrast, constraints are immutable aspects of the design that can be judged on a pass/fail basis. A design that fails to satisfy even one of the constraints must be excluded from the design space. To help define a comprehensive set of objectives and constraints, the design team utilized an objectives/constraints tree, which can be seen in Fig. 3 below [1].

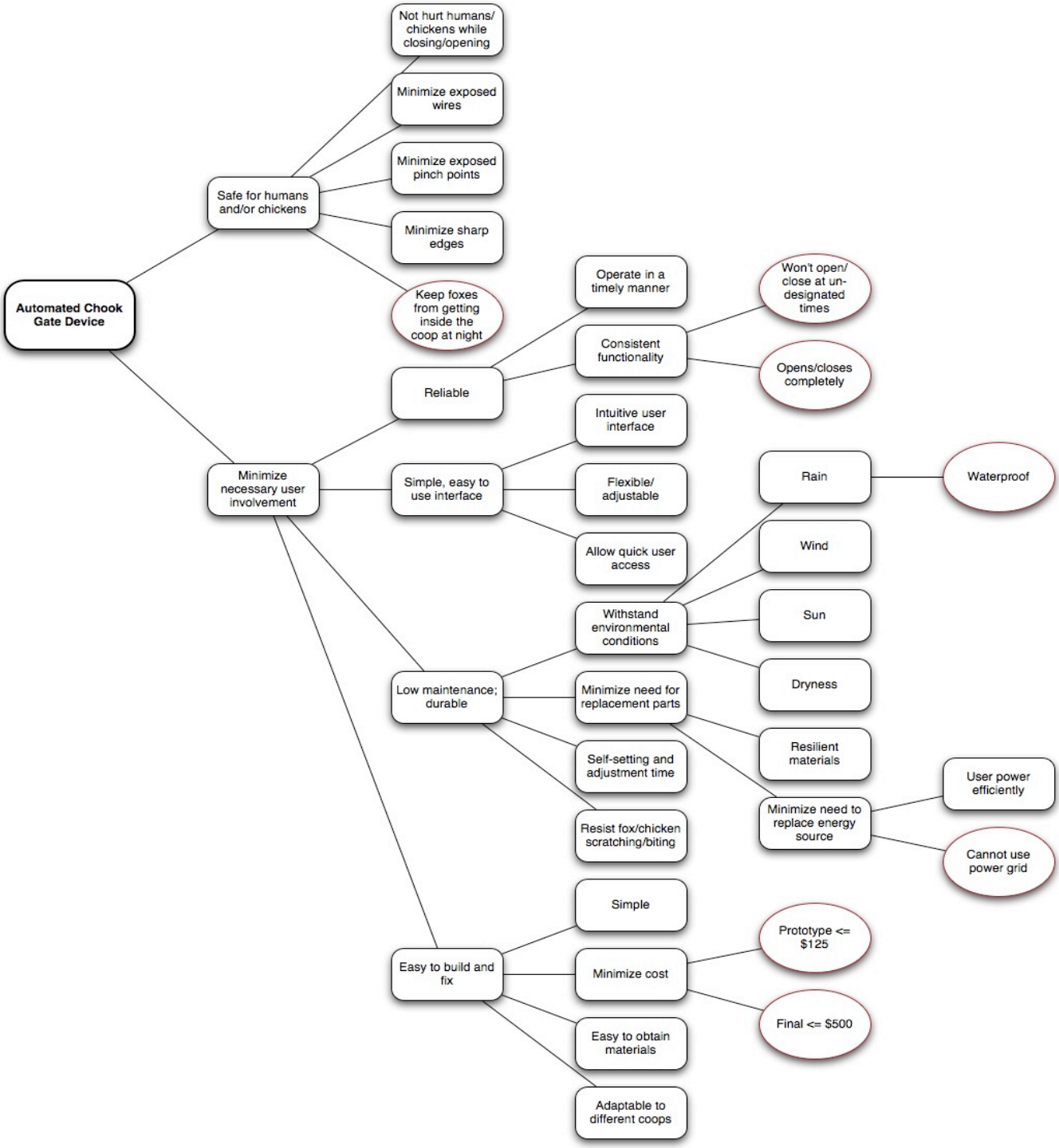


Figure 3: Objectives and constraints tree. Lower level (closer to the right of the page) objectives/constraints accomplish higher level (closer to the left of the page) functions. Constraints are in circles, and objectives are in rectangles.

The team specified metrics, which are measurable aspects of designs that the design team could use to evaluate the design's compliance to the objectives and constraints. Metrics for objectives are quantities or qualities that may be optimized towards a certain state in order to make the design better, while metrics for constraints are thresholds of design parameters beyond which the design does not comply with the constraint. The design team's objectives and metrics are listed in Table 2 below, and the constraints are in Table 3.

Objective	Explanation	Metric
Reliable timing and functionality	The gate needs to open and close consistently at the times designated by the client.	Number of possible ways for unexpected factors to impede the device and their probabilities. The fewer the better
Resist animal forces	The gate needs to resist any forces imposed upon it by animals like chickens and foxes.	Amount of pressure the gate can withstand before getting pushed or broken open. The greater, the better.
Withstand environmental conditions	As the gate will be exposed to the environment at all times, it should be able to withstand dirt, water, wind and any other environmental factors that it might encounter in Ilfracombe, Australia.	The severity of all possible ways the elements (wind, dust, rain, sun) can influence the functionality of the design. The less severe, the better.
Safe for human and chicken use	The device should not itself harm the chickens, and should be safe for any human who is using it.	Number of exposed wires, pinch points, and sharp edges. The fewer the better.
Minimize need for replacement parts	The design should be durable, so that the client does not have to replace parts frequently.	How long the materials and components used are designed to last. The longer, the better.
Easy to fix	The device should be simple enough that fixing the gate does not require entirely disassembling it.	The number of moving parts and ease of accessibility to each part. The fewer moving parts and the greater the ease, the better.
Minimize need for user input (in timing mechanism)	The design should not require extensive user input to keep the gate working satisfactorily to the user.	The estimated frequency the user would have to adjust the timing mechanism. The lower, the better.
Easy to build/adapt to different coops	The design should be easy to replicate from the design team's blueprints and transferable to different chicken coops.	The availability and number of necessary parts. The more available, the better.
Easy-to-use interface	The design should allow the user to adjust the time of opening and closing easily.	The relative ease that a first-time user can figure out how to operate the device. The easier, the better.

Table 2: This table lists the objectives determined by the design tea along with a description of each objective and the metric that the design team used to evaluate it.

Constraint	Pass/Fail Requirement
Waterproof	No functionality may be lost at any point of the device's lifetime due to at least a minute of spraying water as strong as a downpour.
Cannot use power grid	The device must be self-powered (cannot require a plug into an external power grid).
Will not open/close at un-designated times	The device opens only at times designated by the user.
Opens/Closes completely	Door closes sufficiently that there is not greater than 1 inch of open space.
Prototype <= $125	The prototype cannot cost more than $125
Final design <= $500	The final design cannot cost more than $500 to build.
Keep foxes from getting inside the coop	Once closed, the gate can have gaps of no greater than 1 inch.

Table 3: Constraints and their pass/fail requirements.

To determine the order of importance among the objectives, the design team used a pairwise comparison chart, which compares each objective against the others, as seen in Table 4. This was accomplished by listing all of the objectives vertically down the left side of the chart (the y axis) and also horizontally along the top of the chart (the x axis). The team then went through each row of the table and assigned a 1 to boxes below less important objectives, and a 0 to boxes below more important objectives. i.e. a 1 corresponded to the objective on the y axis being more important than the objective on the x axis, and a 0 corresponded to the objective on the y axis being less important than the objective on the x axis. In addition, 'X' was used to denote a table entry where an objective would be compared to itself. Finally, when all objectives had been compared, the design team added up the numbers (only 1's and 0's) across the rows of the pairwise comparison chart. The objective with the highest total was the most important objective (as it was given the most 1's), and the objective with the lowest total was correspondingly the least important objective. The purpose of establishing this ranking of design qualities was to allow the team to decide among design alternatives. In essence, this comparison allowed the team to weight its evaluation of design alternatives towards those designs that fulfill more important objectives. This enabled the team to more accurately determine which designs will be able to meet the client's needs.

	Easy-to-use interface	Withstand environmental conditions	Minimize need for replacement parts	Minimize need for user input	Resist animal forces	Reliable timing and functionality	Easy to build/adapt to different coops	Easy to fix	Safe for human and chicken use
Easy-to-use interface	X	0	0	0	0	0	0	0	0
Withstand environmental conditions	1	X	1	1	0	0	1	1	0
Minimize need for replacement parts	1	0	X	1	0	0	1	1	0
Minimize need for user input	1	0	0	X	0	0	1	0	0
Resist animal forces	1	1	1	1	X	0	1	1	1
Reliable timing and functionality	1	1	1	1	1	X	1	1	1
Easy to build/adapt to different coops	1	0	0	0	0	0	X	0	0
Easy to fix	1	0	0	1	0	0	1	X	1
Safe for human and chicken use	1	1	1	1	0	0	1	0	X

Table 4: The pairwise comparison chart utilized by the design team in ranking their objectives. A 0 indicates that the objective in the column is more important, a 1 indicates that the objective in the row is more important, and an X is used where an objective would be compared to itself.

The pairwise comparison chart resulted in a ranked hierarchy of objectives. The numbers in each row were summed to make a total score for each objective. Objectives with higher scores received higher rankings. Leaving ties with the same rank, the hierarchy was as follows:

1) The design should be reliable in its timing and functionality.
2) The design should resist animal wear.
3) The device should be safe for human and chicken use.
3) The device should withstand environmental conditions.
4) The device should minimize the need for replacement parts.
4) The device should be easy to fix.
5) The device should minimize the need for user input (particularly in the timing mechanism).
6) The device should be easy to build and easy to adapt to different coops.
7) The design should have an easy-to-use interface.

Reliability is the most important objective because the primary purpose of the automated chook gate is to allow the client to leave the device alone for weeks at a time without checking that the door closes every night and that it opens at the right time during the day. If the final design is not able to open and close reliably, other objectives become immaterial because the device no longer fulfills its primary functionality. For example, the design's ability to resist animal forces is no longer significant if the gate simply does not close one night and allows foxes to attack the chickens.

The second most important objective for the designs is to resist wear and tear from animals — both foxes and the daily traffic of chickens in and out of the coop. If the gate is unable to do this, the lower ranked objectives become less important, as foxes are able to invade the coop and kill the chickens. Animal wear resistance is ranked lower than reliability because worn out parts may be replaced, whereas reliability problems are permanently built into the system.

The importance of safety for both humans and chickens follows the previous argument. The chickens will be exposed on a daily basis to a moving gate, which has the potential to hurt them with possible rotating parts, pinch points, sharp edges, etc. A design that does harm to the chickens defies the basic principle of keeping them safe.

The next five objectives follow the overarching theme of minimizing user involvement — that is, reducing the amount of time the user must spend fixing, adjusting, and building the final design. The first of these, "withstand environmental conditions," refers to an alternative's ability to function in the Australian outback where wind, dust, sun, and water (minimally) will reduce the design's lifetime and durability. Following naturally from this is the objective of minimizing the need for replacement parts and making the design easy to fix. This is ranked below the fourth objective because if a design can withstand environmental conditions, it will not need to be fixed as frequently. Minimizing user timer input is ranked at the next level because while it is an important quality, any automation at all is an improvement over daily manual opening and closing. Adaptability and ease of construction come next, since functionality is more important. Also, while it would be optimal to have a portable design, it is only going to be used in one client's coop. Finally, an easy-to-use interface was the lowest rated objective, largely because the design team decided that functionality was, again, more important than ease of use. Particularly, if the final design does not require much user adjustment, the interface will be used as little as possible.

A lower-ranked objective, though considered to be less important, will nonetheless be considered and optimized in the final prototype. This ranking simply means that the team will attach more importance to a design that, for example, is reliable in its timing and functionality but has a poor user interface than to a design with a great user interface that opens unreliably and at the wrong times. The pairwise comparison chart that led to the ranking was merely a tool that helped the team weight their objectives to ensure that the design alternative that was chosen in the selection process was the one that best suited the client's priorities.

4. DEVISING DESIGNS

After completely establishing the design space, the team brainstormed and developed possible designs. This involved coming up with methods or means to accomplish each individual function, synthesizing these means into various design alternatives, and describing the best of these alternatives in enough detail so they could be decided among later. During this process, the team also decided to divide the design into three separate subsystems.

4.1 Morph Chart

Before coming up with designs for complete solutions, the design team designed components to perform all of the functions individually with the aid of a morph chart, as shown in Table 5. It is a table that matches functions to a variety of means to accomplish them. Functions are listed vertically and means to fulfill those functions are listed horizontally. Alternatives can be created from a morph chart by combining one mean from each row into a viable design that executes all of the functions.

Function	**Means**					
Release from open/closed position	Latch	Piston compresses	Move obstacle	Nothing (just opening/closing device starting)	Heat expansion	Solenoids
Move gate to open/closed position	Gravity	Spring	Pneumatic piston	Worm gear (with motor)	Leadscrew (with motor)	Rack & Pinon
Stop gate at open/closed position	Switch (turns off motor)	Velcro	Hits ground	Latch	Magnets	Brake
Resist forces closing/opening gate	Latch	Tension	Weight (gravity)	Spring	Opening device holds in place	Rigid fixture between gate and coop
Accept input	Timer interface	Numeric pad	Dial	Slide	Up/down arrows	
Store input	Memory	Hard drive	Physical storage			
Determine sunset time	Light sensor	Fixed time interval	Solar panel current	Chicken counter	Chicken roosting detector	
Calculate timing	Measure from sunrise	Measure from sunset	Set cycle			
Keep track of time	Timer	Hourglass	Light sensor	Alarm clock	Chicken-movement cycle	
Collect Energy	Solar panel	Connection to house power	Windmill	Dynamo		
Store energy	Battery	Gravitational potential energy	Spring potential			
Barricade foxes	Original gate	Swinging door	Drawbridge	Guillotine	Sliding door	Catflap

Table 5: The design team's morph chart, which lists several possible means for fulfilling each function.

4.2 Grouping Components into Subsystems

The design team noticed that some groupings of functions in the design space are functionally independent. In other words, the means to accomplish some groups of functions have little to no influence over the means to accomplish others. For example, selecting a particular means to calculate a timing method has no effect on what means would then be viable for moving the gate into an open or closed position. Because of this independence, the team found it beneficial to consider sets of functions separately. These different sets of functions were grouped into three different subsystems: the power, gate, and timing subsystems, as shown in Table 6.

Subsystem	Functions
Power	• Collect energy • Store energy
Gate	• Release from open/closed position • Move gate to open/closed position • Stop gate at open/closed position • Resist forces closing/opening gate • Barricade foxes
Timing	• Accept input • Store input • Determine sunset time • Calculate timing • Keep track of time

Table 6: Independent sets of functions separated into three different subsystems.

The power subsystem encompasses all the elements in the design that relate to harnessing energy from a source outside of the device. The gate subsystem involves the set of parts that physically actuate in order to allow or prevent animal transfer to and from the chicken coop. The timing subsystem contains everything that controls when the gate would open or close.

With the complex set of functions broken down into simpler subsystems, the team started the process of separately brainstorming designs for each subsystem. They designed the separate subsystems with the end goal of combining the best design in each subsystem into a full, final design.

4.2.1. POWER SUBSYSTEM

It was important to the client that the device not be connected to a wall outlet, as this would require an unwieldy cable and may increase the electricity bill by a considerable amount. This meant that the device had to harness external power in order to operate. The most apparent solution, and the one that was supported by the client, was to use solar energy. After considering many other sources of power, the design team decided that sunlight would be the most abundant, reliable, practical, and inexpensive source of energy for the device.

After some research regarding solar energy, the design team discovered that a solar panel would be capable of powering the device. The design team estimated that the opening and closing processes would each take a maximum time of one minute, and that the gate would only open once and

close once each day. Therefore, the net wattage (energy per unit time) of the gate's operation should be small. Additionally, the device is located where there is an abundance of sunlight in Ilfracombe, Australia) and subsequently plenty of solar energy. The design team concluded that a 1 watt photovoltaic cell would be adequate for powering the device.

Using a 1 watt solar panel requires a method of electric potential storage, as the low output would not be able to directly drive the system. The design team decided to use a voltaic cell (a battery) since it is the most common and available method for storing electrical energy. The team found that most small electrical devices are run on 12 Volts of potential, so they chose that as their battery voltage. This meant that the solar panel would be rated at 12 Volts as well [11].

4.2.2 GATE SUBSYSTEM

As mentioned in section 4.2, the gate subsystem encompasses the parts of the design that physically open and barricade the chicken coop. The gate includes the barrier between the chickens and the outside world. Furthermore, it includes the device that moves the barrier (such as a motor with a gear attached).

Design Alternatives

To devise designs for the gate subsystem, the design team initially used the morph chart (Table 5) to combine means to accomplish each function (Table 1). Many different concepts and design alternatives emerged from this process. Following the brainstorming, the team narrowed down their collection of designs to the four most viable ones.

The four alternatives that the design team considered for the mechanical portion of the design were the push latch design, the garage door, the sliding door, and the small swinging door. Each of these designs are powered by motors coupled with apparatuses that will not backdrive. In other words, the drive system for the barricade prevents the barricade from being moved by outside forces. There are two of these apparatuses among the design alternatives: a leadscrew and a worm gear.

A leadscrew is a specialized screw that translates rotational motion (in this case, the spinning of the motor) into linear motion (the translational movement of the gate). Threaded "riders" are threaded onto the leadscrew, just like nuts are threaded onto bolts. The riders are forced to maintain a certain orientation such that when the screw turns, they are pushed along the length of the screw.

4.2.2.1 The push-latch design

Of the four final design alternatives, the push-latch design is the only one that uses the pre-existing gate to the chicken coop. It involves a motor-driven leadscrew to push and hold the gate open, a spring to pull the gate shut, and a latch that keeps the gate closed.

This design utilizes a leadscrew mounted at the top of the client's existing gate to open the gate for the chickens. The leadscrew would have a plate "pusher" that would serve to both push the gate open and compress a latch to release the gate from its locked position. The leadscrew would also stretch a spring that would be attached to the door, storing spring potential energy to pull the door back when needed. The leadscrew would be powered by a motor that would run in one direction to open the door, then be reversed to allow the spring to close the gate and latch it shut. This design is diagrammed and labeled in Fig. 2 below. This design uses the following means to fulfill each function from the morph chart:

- Release from closed position - plate pusher compresses push bar
- Move gate to open position - motor rotates leadscrew with plate pusher on the end that pushes the door open
- Stop gate at open position - limit switch at the end of the extension

- Resist forces closing gate - the leadscrew won't backdrive
- Release from open position - the motor would rotate the other way bringing the plate back and letting the spring force bring the gate back in
- Move gate to closed position - the extended spring would pull the gate back
- Stop gate at closed position - pushbar latched again
- Resist opening forces (foxes) - the pushbar is latched and cannot be opened from the outside
- Barricade foxes/chooks (Door type) - client's original gate

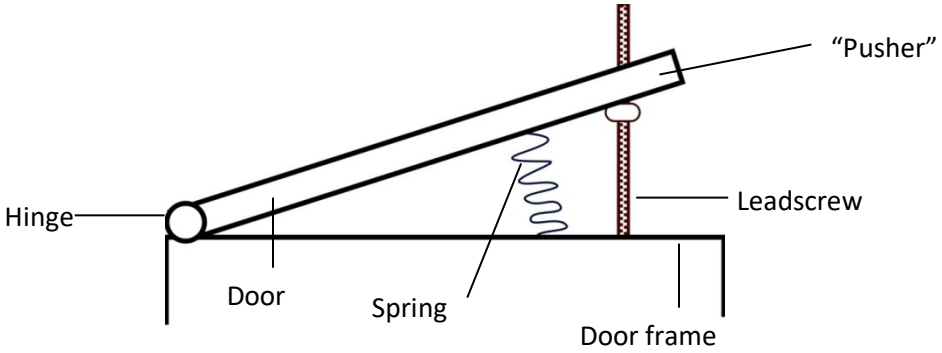


Figure 4: This figure illustrates the push-latch design alternative, with each significant component labeled.

In order to determine the necessary output torque of the motor used for this design alternative, the design team utilized physical models of the gate and the opening system. First, the team let the leadscrew be a minimal distance from the opening edge of the door. However, as the door opens, the nut on the leadscrew effectively moves towards the opening edge of the door. Thus, the leadscrew must be placed a minimum distance of *d* from the door's edge, so that the leadscrew maintains contact with the door until it opens to a specified distance *s*. The geometry of this design is illustrated in figures 5 and 6 below.

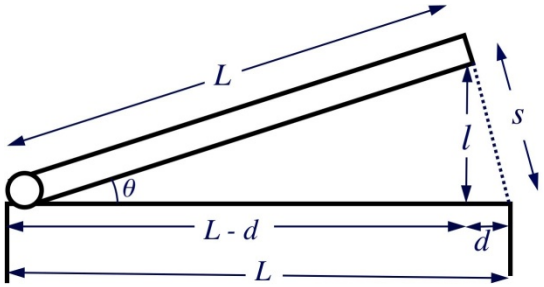


Figure 5: This figure demonstrates the geometry of the push-latch door.

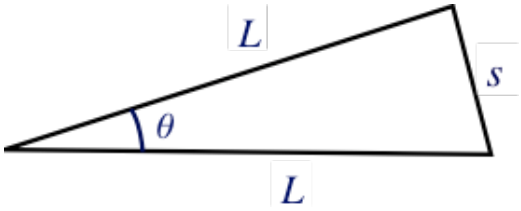


Figure 6: The door can be viewed as an isosceles triangle with two sides of length L and a base of length s.

As figure 4 illustrates, the gate can be viewed as an isosceles triangle with a base of length *s*. The design team used equation (1), the law of cosines, to derive an expression for the angle between the gate and the door frame, so that this expression could later be used to find the minimum value for the distance *d* of the leadscrew from the edge of the gate.

$$s^2 = L^2 + L^2 - 2LL\cos\theta \quad (1)$$

$$s^2 = 2L^2(1-\cos\theta) \quad (2)$$

$$\frac{s^2}{2L^2} = 1 - \cos\theta \tag{3}$$

$$\cos\theta = 1 - \frac{s^2}{2L^2} \tag{4}$$

When the leadscrew is placed at the minimum value of *d* as shown in figure 5, the gate can be viewed as the hypotenuse of a right triangle with side lengths of *L*, *L-d*, and *l*. This is illustrated in figure 7 below. Trigonometric formulas may then be used to solve for *d* in terms of the length *s* that the client wants the door to open and the length *L* of the door itself.

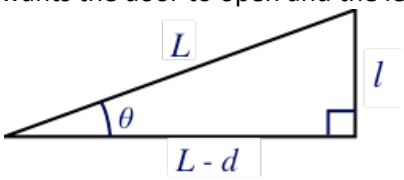


Figure 7: When the leadscrew is at the minimum distance d, the door can be viewed as a right triangle.

$$\cos\theta = \frac{L-d}{L} \tag{5}$$

$$d = L - L\cos\theta \tag{6}$$

$$d = L - L\left(1 - \frac{s^2}{2L^2}\right) \tag{7}$$

$$d = L - L + \frac{s^2}{2L} \tag{8}$$

$$\boxed{d = \frac{s^2}{2L}} \tag{9}$$

The necessary length of the leadscrew can also be determined using physical modeling. The length of the leadscrew will vary as a function of *d*, the leadscrew's distance from the edge of the gate. Using trigonometry, one can find an expression for the length of leadscrew by again viewing the gate as a right triangle, as shown in figures 8 and 9 below.

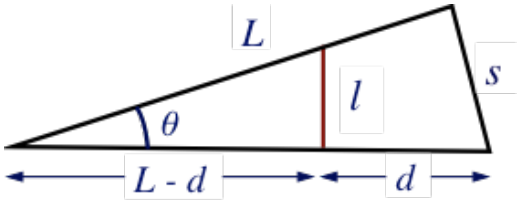


Figure 8: This figure illustrates the geometry of the leadscrew's positioning and how this relates to its necessary length.

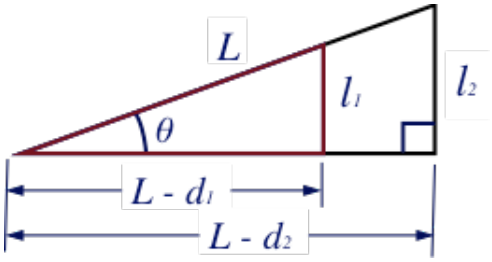


Figure 9: The necessary length of the leadscrew varies based on its position along the door frame, but its length can be determined by modeling the gate as a right triangle.

As can be seen in the diagrams above, *l* is related to *L* and *d* through the tangent of the angle of the gate's opening.

$$\tan\theta = \frac{l}{L-d} \tag{10}$$

$$l = (L-d)\tan\theta \tag{11}$$

The previously calculated value for the cosine of the gate's opening angle can then be used to give a final expression for *l*.

$$\cos\theta = 1 - \frac{s^2}{2L^2} \tag{12}$$

$$\theta = \cos^{-1}(1 - \frac{s^2}{2L^2}) \tag{13}$$

$$\boxed{l = (L-d)\tan(\cos^{-1}(1 - \frac{s^2}{2L^2}))} \tag{14}$$

When considering the forces acting on the door, the design team chose to consider the door's pivot point as the axis of rotation, as shown in figure 10. The two forces causing torques on the door are then the force from the leadscrew pushing the door open, and the force from the spring pulling the door back.

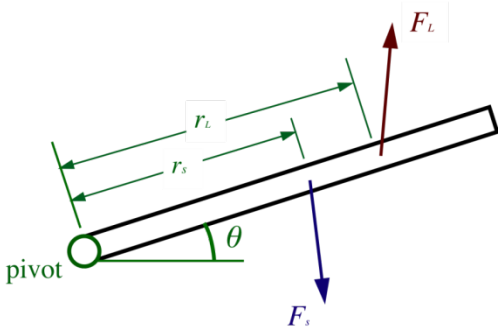


Figure 10: The two forces acting on the door are the force from the leadscrew and the force of the spring.

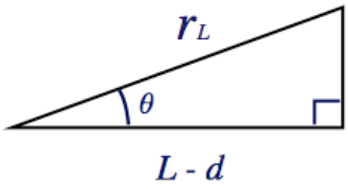


Figure 11: Once more, the door can be viewed as a right triangle.

By again viewing the door as a right triangle, as in figure 11, the design team found an expression for r_L.

$$\cos\theta = \frac{L-d}{r_L} \tag{15}$$

$$r_L = \frac{L-d}{\cos\theta} \tag{16}$$

The gate can also be viewed as an isosceles triangle, as in figure 12, and the resulting geometry gives expressions for how far the door has opened and for the other angle in the triangle, Φ. This in turn helps to determine the components of the forces acting on the door, which assists in the calculation of the torques these forces impart upon the door.

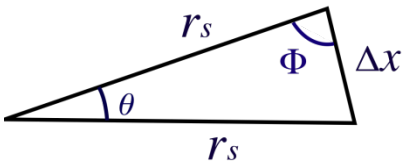


Figure 12: The door can be viewed as an isosceles triangle to assist in determining the perpendicular and parallel components of the forces.

$$\frac{\Delta x}{r_s} = \sin\theta \tag{17}$$

$$\Delta x = r_s \sin\theta \tag{18}$$

$$\Phi = \frac{180 - \theta}{2} \tag{19}$$

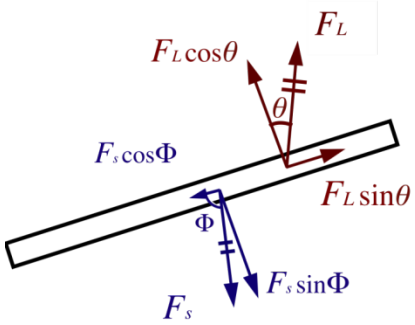


Figure 13: The forces acting on the door have perpendicular and parallel components that are necessary for modeling the torques that these forces create.

$$\sum \tau = I\alpha \tag{20}$$

The sum of the torques generated by the spring force and the force of the leadscrew are equal to the gate's moment of inertia multiplied by its angular acceleration, as shown in equation (20).

In order to be an effective design, the door must have an angular acceleration that is much greater than 0. The design team used this assumption to solve for the force that the leadscrew must impart upon the gate as a function of the angle to which the door has been opened.

$$r_L \times F_L - r_s \times F_s = I\alpha >> 0 \tag{21}$$

$$\frac{L-d}{\cos\theta} F_L \cos\theta - r_s k r_s \sin\theta \sin\Phi >> 0 \tag{22}$$

$$F_L >> \frac{r_s k r_s \sin\theta \sin\Phi}{L-d} \tag{23}$$

$$F_L >> \frac{r_s^2 k \sin\theta \sin(\frac{180-\theta}{2})}{L-d} \tag{24}$$

This force can then be used as the load, *P*, in an equation for the necessary torque output of the leadscrew. This output can, in turn, be assumed to be the output needed from the motor on which the leadscrew is mounted. Equation 25, below, models the torque required to lift a load vertically with a leadscrew [7].

$$T_{up} = \frac{Pd_p}{2}\left(\frac{\mu\pi d_p + L\cos\alpha}{\pi d_p \cos\alpha - \mu L}\right) \quad (25)$$

In this equation, *P* is the load being lifted, d_p is the thread diameter, *L* is the lead (inches per thread), α is the thread angle, and μ is the coefficient of friction between the riders and leadscrew [Machine Design 822]. In order to calculate the necessary torque of the motor, one need only substitute values for each of the unknowns in the formula, using the calculated value of F_L as the load.

4.2.2.2 The Garage Door Design

This design also uses a leadscrew, but it is instead mounted on the side of a 'garage door' style gate. The leadscrew would have two 'riders' threaded through the rod and connected to the gate at its bottom and midpoint so that when the motor turns one way, it will force the two riders up the leadscrew and the door with them. Similarly, the motor will turn the other way to let gravity take the gate and riders back down to the closed position. A limit switch will be placed at the 'open' and 'closed' positions, so that power to the motor will be cut and the gate's motion will be stopped. This design's means for fulfilling each function in the morph chart are as follows:

- Release from closed position - nothing, just start raising the gate
- Move gate to open position - leadscrew and riders
- Stop gate at open position - motor stops rotating when limit switch is hit and then the leadscrew does not backdrive so can rest in its open position all day
- Resist forces closing gate - sealed within the gate itself, and leadscrew does not backdrive
- Release from open position - motor turns in opposite direction
- Move gate to closed position - leadscrew forces gate riders down the screw thus pushing the door down also
- Stop gate at closed position - the floor or a groove in the floor is hit (limit switch)
- resist opening forces (foxes) - weight of the door holds it in place and the leadscrew will not backdrive up the shaft
- Barricade foxes/chooks (Door type) - garage-door style door

There was much less to model in this alternative as compared to the push-latch design. The design team only needed to model the torque required by a motor to rotate a leadscrew with certain specifications (i.e. diameter, threads per inch, frictional coefficient) to get an idea of how much torque and what kind of motor the garage door design required and decide if it was a feasible alternative.

The torque required to lift a load vertically is the same as given above in equation (25) and although it appears daunting at first, there are several quantities known and the others are given by the specifications of the leadscrew. For example, the design team planned on using an Acme threaded screw; these always have a thread angle α equal to 14.5°. Additionally, the team used the worst case scenario for the coefficient of friction between the riders and leadscrew,$\mu = 0.25$, to ensure that the required torque was not underestimated.

Now the only variables left are the load *P*, the thread diameter d_p, and the lead of the screw, *L*. All of these are quantities given by the team's choice in the gate and leadscrew to be used. As a result, the team researched leadscrews and determined that for the chook gate project, a ½" leadscrew with a lead of 10 threads per inch would be adequate to ferry an approximately 7lb gate up and down. Plugging these numbers into the above equation yields

$$T_{up} = \frac{(7lb)(0.5in)}{2}\left(\frac{(0.25\pi(0.5in)+(\frac{1}{10})\cos(14.5)}{\pi(0.5in)\cos(14.5)-0.25(\frac{1}{10})}\right) \quad (26)$$

$$T_{up} = 0.155\ ft \cdot lbs$$

This would be the maximum torque a motor would have to output in order to raise the gate and 0.16 ft lbs of torque is feasible for a small motor to produce [7].

4.2.2.3 The Sliding Door Design

This designs involves using a small, chicken-sized (1' x 2') sliding door gate mounted on the client's original gate. A leadscrew mounted above the door would be attached to the gate with two leadscrew 'riders' that move to the left or the right based on which direction the motor is rotating. As the motor turns one way, it will push the two riders across its threads, and thus pull the gate left or right with it until the gate hits a limit switch at its open position that signals the stop of the motor. This design's means for fulfilling each function are as follows:

- Release from closed position - the motor begins rotating
- Move gate to open position - leadscrew and riders
- Stop gate at open position - motor stops rotating when limit switch is hit and then the leadscrew doesn't backdrive so can rest in its open position all day
- Resist forces closing gate - sealed within the gate itself, and leadscrew does not backdrive
- Release from open position - motor turns in opposite direction
- Move gate to closed position - leadscrew forces gate riders in the opposite direction and so pushes door closed
- Stop gate at closed position - gate activates a limit switch at its closed position
- resist opening forces (foxes) - the leadscrew would not backdrive
- Barricade foxes/chooks (Door type) - sliding door

The modeling of this design largely follows the modeling of the garage door design because both use a leadscrew to move the door from its open to closed positions. The only difference between the two is that the garage door uses a vertical leadscrew and thus has a load of the entire gate whereas the sliding door uses a horizontally-oriented leadscrew. As a result, the load on this leadscrew would be given by the force needed to accelerate the gate in addition to the frictional force between the bottom of the gate and the ground, as given in equation 27.

$$F_{net} = ma + F_f \quad (27)$$

The dimensions of this sliding door can be estimated as 18 inches by 18 inches, so if the gate were to open completely in 15 seconds, its acceleration would be determined by equation 28:

$$\Delta x = \frac{1}{2}at^2 + v_o t + x_o \quad (28)$$

where Δx represents the distance the door opens, a is acceleration, t is time, v_0 is the initial velocity and x_0 is the door's initial position. Given an initial velocity of zero, an initial position of zero, and solving for acceleration,

$$a = \frac{2\Delta x}{t^2} \quad (29)$$

$$a = \frac{2(1.5 \text{ ft})}{(15 \text{ sec})^2} = \frac{3}{225}\frac{ft}{sec^2} = 0.013\ \frac{ft}{sec^2}$$

The frictional force can then be calculated by equation (30),

$$F_s = \mu_s N \tag{30}$$

where μ_s is the coefficient of static friction, which can be approximated as 0.5, the coefficient of friction between wood and metal [9]. With the same seven pound gate as used to model the garage door design, then, the net force required to move the gate would be 7lb (0.5) + 7lb (0.013 ft/sec^2) = 4 pounds. Using equation 1 above with the load, *P*, equal to 4 pounds and all other variables held constant (i.e., calculated using the same leadscrew specifications), the torque required to move the gate back and forth would be approximately 0.09 ft pounds.

The above calculation is purely approximate. It is very loose and dependent on the approximations made including acceleration of the gate, frictional forces, materials used in a final prototype, etc. The design team attempted to model each of the designs as best as possible to get a very general idea of the motor necessary for each of the designs and also to estimate the viability of each of the alternatives.

4.2.2.4 The small swinging gate design

This design involves using a small swinging gate (1' x 2') attached to the front of the client's gate. This would be powered by a motor attached to a worm gear at the hinge and would swing out horizontally. The means used by this design to fulfill each necessary function are as follows:

- Release from closed position - the motor starts turning
- Move gate to open position - the motor turns a specific number of rotations
- Stop gate at open position - motor only spins for a specific amount of time
- Resist forces closing gate - the wormgear will not backdrive
- Release from open position - not necessary-the motor rotates the other way
- Move gate to closed position - the motor rotates the other way and thus pushes the worm in the opposite direction of rotation
- Stop gate at closed position - limit switch
- resist opening forces (foxes) - wormgear doesn't backdrive
- Barricade foxes/chooks (Door type) - miniature swinging gate

The design team used physical modeling to determine the torque the motor connected to the worm gear would have to produce. The output torque of a worm gear can be modeled using equation (31),

$$M_2 = (M_1 - \frac{d_2}{d_1}) \cdot \frac{\cos\alpha_n - \mu\tan\gamma}{\cos\alpha_n \cdot \tan\gamma + \mu} \tag{31}$$

where M_2 is output torque, M_1 is the input torque, d_2 is the pitch diameter of wormwheel, d_1 is the pitch diameter of worm, α_n is the normal pressure angle, μ is the coefficient of friction, and γ is the worm lead angle [6]. In order to estimate the value of M_1, the design team approximated the door as a solid rectangle with a homogeneous distribution of mass. This allowed the team to estimate

$$I = \frac{1}{3}ML^2 \tag{32}$$

where M is the mass of the door and L is the width of the door. The necessary output torque, then, is once more equal to the moment of inertia multiplied b the angular acceleration.

$$\sum \tau = I\alpha \tag{33}$$

The design team then made several estimations and approximations based off of standard values and used these to estimate the value of M_1. The design team estimated:
$\alpha = 0.5$ rad/s^2
M = 3 kg
L = 0.3 meters
These estimations in turn gave M_2 as 0.45 Nm. The team also estimated that d_2 = 1.5 inches (0.0381 m), d_1 = 1 inch (0.0254 m), $\alpha_n = 14.5^\circ$, $\mu = 0.78$ and $\gamma = 12^\circ$. Using these numbers in equation (6) above gives M_1 = 0.37 Nm, or 0.27 ftlbs [4][5].

The design team used the results of their physical modeling to determine that all of the design alternatives were indeed feasible, as well as in their evaluations during the design selection process.

4.2.2.5 Selection Process

The design team decided upon the best of class method [1] to evaluate their four design alternatives and select their final design. The best of class method involves creating a table listing all of the design objectives vertically and the four design alternatives horizontally. The team compared the four alternatives for each objective, ranking each on their adherence to the metrics. Table 7, below, shows the results of the design team's evaluation of the design alternatives.

Objective	Vertical Leadscrew	Horizontal Leadscrew	Swinging Door	Push-latch Door
Reliable timing and functionality	1	2	3	4
Resist animal forces	1	2	3	4
Withstand environmental conditions	1	2	3	4
Safe for human and chicken use	4	2	3	1
Minimize need for replacement parts	1 (tie)	3	1 (tie)	4
Easy to fix	2 (tie)	2 (tie)	1	4
Minimize need for user input (in timing mechanism)	1	1	1	1
Easy to build/adapt to different coops	1	2	3	4
Easy-to-use interface	1	1	1	1

Table 7: This table shows the results of the best of class evaluation of the design alternatives. Two of the objectives concern the timing system of the design, resulting in a four-way tie since all of the design alternatives utilize the same timing system. Because the "Minimize need for user input" and "easy-to-use interface" objectives are not influenced by the gate subsystem designs (they are involved in the timing subsystem), they all received the same score of 1.

For the first and most important objective, reliable timing and functionality, the design team came to the conclusion that the vertical leadscrew design would be best, followed by the horizontal leadscrew, the swinging door and finally the push-latch door. The vertical leadscrew design would be the least likely to jam and stop functioning, since gravity would help disperse any dust that began to accumulate in the threads of the leadscrew. Meanwhile, both the swinging door and the push-latch door would spend several hours each day sticking out, susceptible to wind gusts and other environmental factors. The leadscrew used in the push-latch door would also stick out all day, leaving it open to more damage and thus increasing the risk of decreased functionality.

For the second objective, resisting animal forces, the vertical leadscrew was once again deemed the best, the horizontal leadscrew second, and the swinging door and push-latch third and fourth respectively. The weight of the vertical leadscrew door holds it closed very securely, while the small

space at the bottom of the horizontal leadscrew design leaves it more susceptible to a fox tilting or rotating the door. The push-latch door is only pinned down at one point, so it has the potential of having its latch not close. The swinging door also has the problem of being pinned down at only one point, but its smaller size makes this less of an issue.

The rankings amongst the design alternatives remained the same for the third most important objective, withstanding environmental conditions. The design team determined that the vertical leadscrew design would be the least exposed to dust, wind and other environmental factors, while the horizontal leadscrew design would have more bearings exposed to the environment. The swinging door would be vulnerable to buffeting from the wind and would also be more exposed to dust and dirt due to the slant of the worm gear. As with previous objectives, the push-latch door was considered worst due to its exposed spring and leadscrew.

The next objective, safety for humans and chickens, had very different results from the previous three objectives. The vertical leadscrew design was deemed the worst because of its guillotine-like structure and greater number of pinch points, while the spring-loaded nature of the push-latch door caused it to be ranked highest. Both the swinging door and the horizontal leadscrew designs have few pinch points, but the inability of the leadscrew and worm gear to backdrive caused them to be ranked slightly worse than the push-latch door.

For the fifth most important objective, minimize replacement parts, the vertical leadscrew and swinging door designs tied for first, followed by the horizontal leadscrew design in third and the push-latch door in last. The vertical leadscrew and swinging door designs both require few parts, making them very easy to replace. The horizontal leadscrew design has slightly more parts, as it has wheels in addition to the leadscrew, and the push-latch door requires many more parts than the other designs.

When considering the objective of “easy to fix”, the design team determined that the swinging door best fulfilled this objective, followed by the vertical and horizontal leadscrew designs and with the push-latch design once again in last place. The worm gear that rotates the door in the swinging door design would be fairly simple to replace, since it is small and easily removed, while a three-foot leadscrew would be somewhat more complicated. The push-latch design has many more parts and is much larger, so it would be the most difficult to fix.

The design team also determined that the vertical leadscrew design would be the easiest to build and adapt to other coops because it is fairly simple, while the horizontal leadscrew design requires more precision in measurements. The swinging door design also requires more precision to build and assemble, and the push-latch design has many more parts and is less adaptable.

Overall, the team came to the conclusion that the vertical leadscrew design best fulfilled the objectives, as it consistently ranked first as compared to the other designs. This design only received one last-place ranking, and the team determined that this design, while slightly more dangerous for chickens than the other designs, nonetheless did not pose a significant danger to the chicken’s lives.

4.2.2.6 Final Design and Prototype

As Figure 14 shows, the mechanical subsystem of the final design involves a smaller, chicken-sized door that is affixed via three “riders” to a vertically mounted leadscrew. A motor turns the leadscrew, causing the riders to travel up and down along the length of the rod, and thus the door they are attached to, to move up or down. This motion serves to open or close the gate, allowing the chickens access to the yard during the day and keeping them safe from foxes during the night. The leadscrew is coupled to the motor on one end and rotates freely inside a bearing on the other. The motor and the bearing are both held in place by L-brackets. On the other side of the gate, the door itself slides up and down in a channel. The channel serves to increase the stability of the door, and keeps the gate and riders oriented in the same direction to prevent the door from spinning with the leadscrew. Although this component is not shown in Fig. 14, the final design would also have a waterproof container around the leadscrew and motor. This container would prevent water from hitting the motor and causing it to short circuit, keep dust and dirt from jamming the leadscrew as easily, and would protect humans and chickens from moving parts.

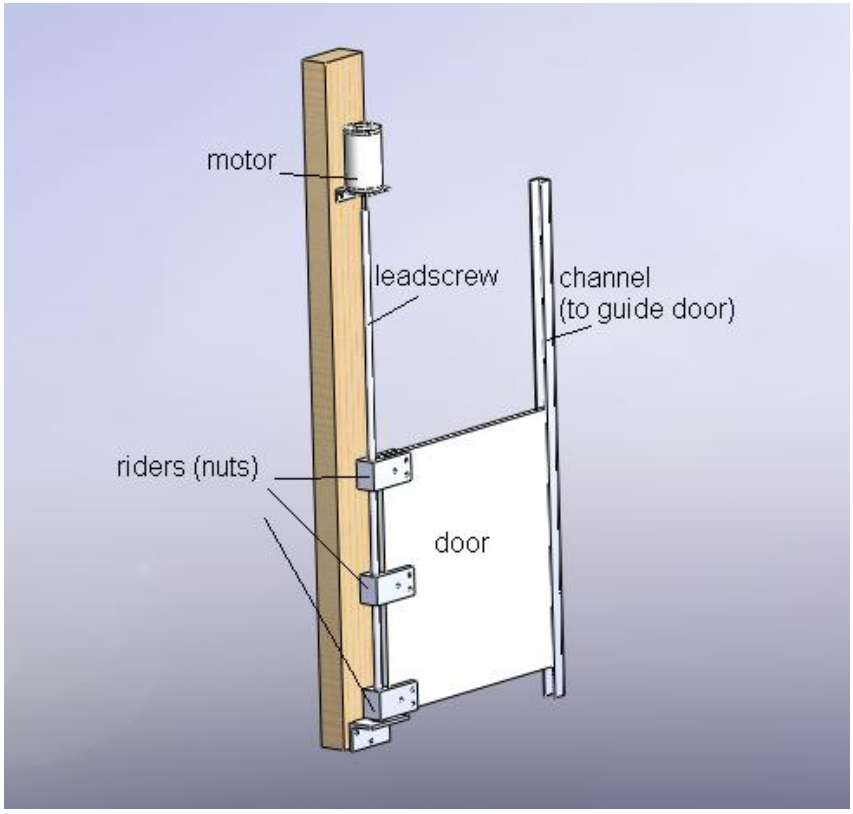


Figure 14: Drawing of final design.

The design team created a prototype of the final design shown in Fig.15. The prototype uses a large sheet of plywood with a 15” by 20” hole cut in it as the main mounting surface for the actual gate. The team also affixed two two-by-fours to the main mounting surface, attaching one to each side of the hole. One of the two-by-fours serves as the mounting area for the motor. The motor is held in place at the top of the two-by-four by a motor mount, with the motor shaft pointing down. Coupled to the motor shaft is a three foot long threaded rod, which runs down the length of the two-by-four to connect to a flanged bearing inside of a bearing mount. The bearing allows the threaded rod to rotate freely while still holding it fixed vertically along the two-by-four. The threaded rod is also connected to the

actual coop door, a 15” by 20” plastic cutting board, by a rider. The cutting board was selected because it was readily available, waterproof, lightweight, and designed to withstand cuts and scrapes on a regular basis. The rider consists of a block of metal with a threaded hole that screws onto the threaded rod and an inset area that the door is fitted and bolted into. When the motor turns, the threaded rod turns as well, moving the rider and thus the door either up or down. The door slides inside of a channel affixed to the other two-by-four, so that friction from the rotation of the threaded rod cannot twist the door outward and allow foxes entrance. The functions of each component of the prototype are given in Table 8 below.

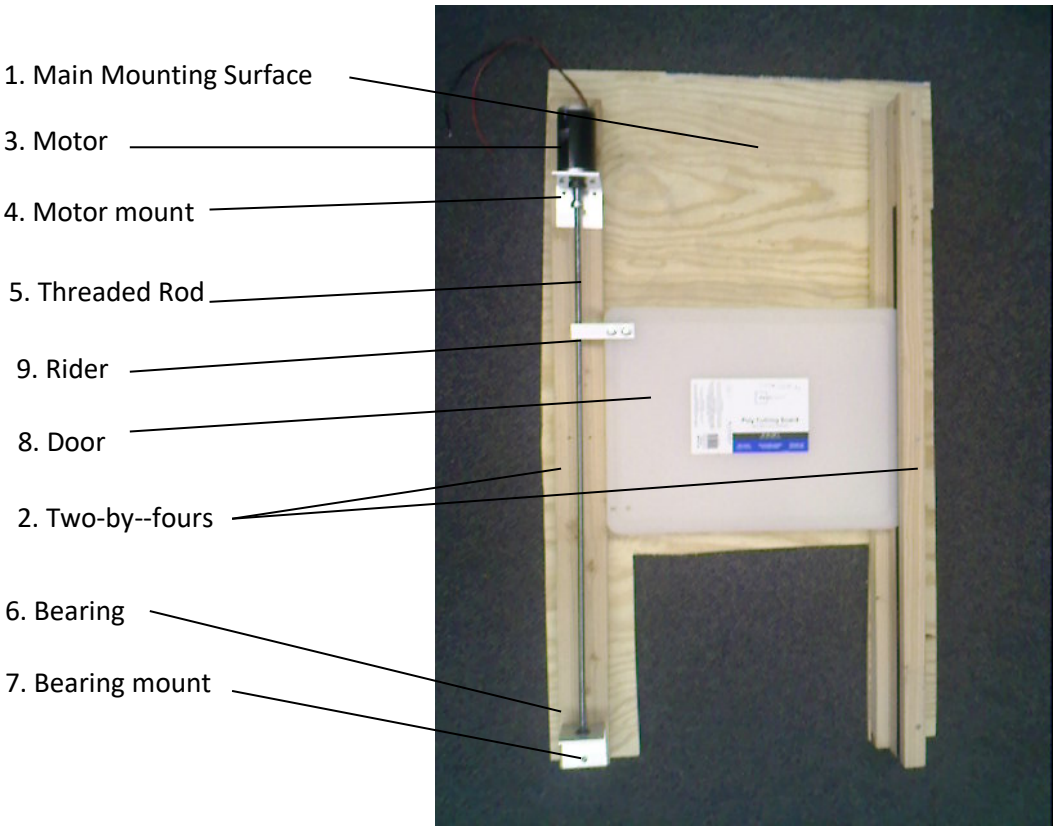


Figure 15: The design team's final prototype.

Part	**Function**
1. Main mounting surface	Support components of door-opening system
2. Two-by-fours	Contain channel; support motor and bearing mounts
3. Motor	Rotate threaded rod
4. Motor mount	Attach motor to two-by-four
5. Threaded rod	Convert rotational motion of motor into linear motion of door
6. Bearing	Hold threaded rod in place
7. Bearing mount	Attach bearing to two-by-four
8. Door	Block entrance to coop
9. Rider	Attach door to threaded rod

Table 8: Functional analysis of mechanical subsystem of the team’s final design.

Basic tests of the prototype had very promising results. Attaching the threaded rod to a hand drill and rotating it successfully moved the rider, and thus the door, up and down as needed. The door was also able to slide easily inside of the channel. The team needed to modify some of their original plans, however, in light of unanticipated problems. When multiple riders were attached to the door, the team had difficulty screwing the threaded rod into the riders. Any skewing of the riders would misalign the threads of the rod with those in the rider, preventing the threaded rod from rotating through the rider as needed.

The team decided upon several recommendations for improvement to the door-opening subsystem of the design when it is actually built. The use of an Acme leadscrew is highly recommended, as these leadscrews are intended for carrying loads and thus will prove both more effective and less fragile than a generic threaded rod. Riders specifically intended for Acme leadscrews can also be purchased. The team also determined that the use of a backup door would improve the design. This door could be manually closed in the event that the door somehow gets stuck open, protecting the chickens until the door can be fixed.

4.2.3 TIMING SUBSYSTEM

The design process for the control system of the Chook Gate was more linear than the process for the mechanical system. Choosing a good design was a matter of following the design constraints to the only logical solution to the problem. Not many formalities were used. The design team considered the ways in which the control system would "know" when to open the gate. As specified by the client, the gate had to open and close at certain times relative to sunset.

4.2.3.1 Timing Logic

The first question that the design team had to answer was whether the control system would be run by a premade microcontroller or a custom built circuit. Microcontrollers are essentially smaller, less powerful computers than average household computers. They have inputs, outputs, and a programmable processor. Microcontrollers are useful because they may be programmed to execute the logic that would otherwise have to be built into a circuit [10]. The design team decided that the logic of the gate's operation was too complicated to wire from scratch. A logic circuit without a microcontroller would be so complicated that it would make building and fixing the device significantly more difficult. Thus, using a microcontroller instead of a custom logic circuit best satisfied the objectives of "easy to build/adapt to different coops" and "easy to fix."

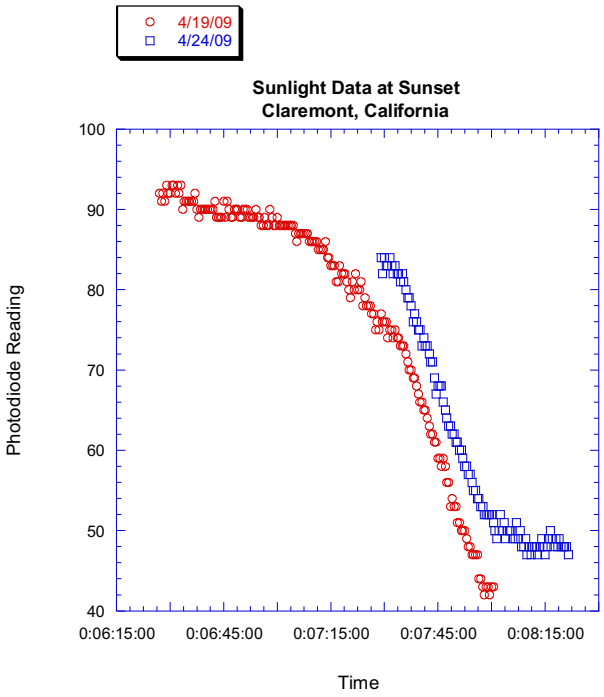
One method of giving the system input for timing its operation was to use only a timer to count 24 hour intervals between sunsets. However, as shown in figure 1, the timing between sunsets is not necessarily 24 hours. Thus, the system would have to adjust the timer to account for varying sunset times. Making this adjustment would be complex. Furthermore, the timer would probably have at least a small amount of error that would accumulate over time. Eventually, the timing of the gate's operation would be off such that it would not fulfill the constraint of "will not open/close at un-designated times." Thus, using a timer alone would not be reliable enough input to control the gate.

The design team decided that sunset would be the best regularly repeating reference for timing the opening and closing of the gate. Thus, the control system would contain a sunset detector and a timer. The sunset detector signals the system at sunset so that the system could recalibrate its timing daily. The timer counts time intervals such that the system could reference the opening and closing time to the sunset signal.

4.2.3.2 Detecting Sunset

The easiest way to detect sunset seemed to be to signal the system whenever ambient light outside darkened below a certain threshold. The design team considered using either a photoresistor or a photodiode to detect ambient light intensity. A photoresistor is a resistor which has a resistance that varies proportionally to the amount of light hitting it. The resistance of the photoresistor would be detected by the microcontroller. The design team discovered that a more available resource for the purposes of prototyping was a photodiode. A photodiode is a miniscule solar cell. It outputs a voltage across its two terminals that is proportional to the amount of light hitting it. The design team used a photodiode for testing the control system.

The design team tested the values that a microcontroller read when the photodiode was used as an input. The data collected are shown in figure 14 and figure 15. The data shown are values read from an analog input port on an Arduino microcontroller plotted with respect to time. The anode of the photodiode was connected to the GND (ground) pin on the Arduino. The cathode of the photodiode was connected to the ANALOG IN 0 pin. The range of values that the Arduino could read on the port used for testing was 0 – 1023. These data sets were taken over time intervals that started in full daylight (between 6 and 7 PM) and ended at night (around 8:30 PM). The blue data set was taken on an overcast day.



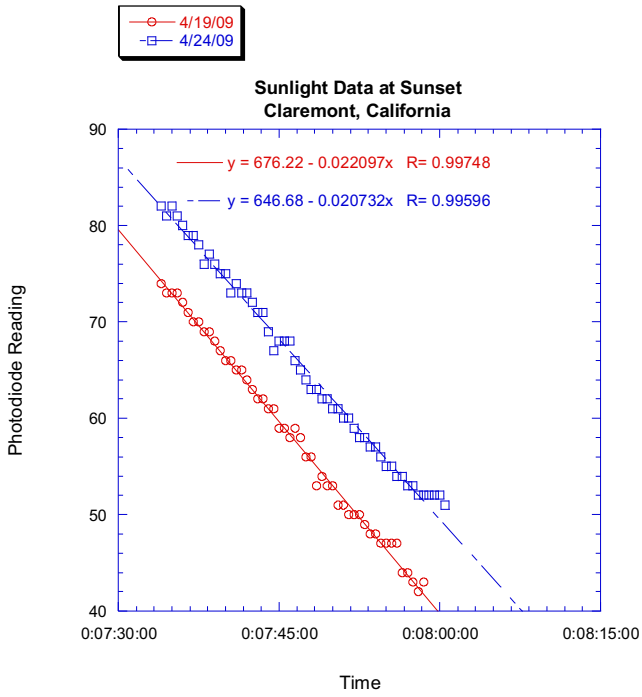


Figure 16: (left): Readings of the photodiode with respect to time over the course of a full sunset.
Figure 17: (right) The most linear portion of the data from fig. 14 and plotted with fit line.

The diagrams show a clear trend in the values obtained as input from the photodiode. As the sun set, the values read steadily declined until they hit a minimum and stopped falling. The steadily reoccurring values after sunset were noticeably lower than those before sunset. Thus, the design team concluded that a threshold value existed past which it could conclusively be stated that sunset had happened.

As an alternative to the threshold value method of detecting sunset from the data, the design team considered using the rate of change in the values with respect to time. As can be seen in figure 15, the slope of the values is fairly constant as the sun sets. Even more promising is the fact that the slope was very similar between the two runs.

The design team was initially concerned about the apparent volatility of the data. If the values were volatile enough, the sensor could prematurely send a signal that was past the sunset threshold and thus disrupt the operation of the device. However, in both tests, readings before sunset were never below readings after sunset, so the threshold could still be established.

There was some ambient light that was not emitted by the sun but by city lights surrounding the test location that turned on when the sun began to set. This may have raised the post-sunset values by a constant amount. Thus, the difference between readings before and after sunset should be greater in the rural location where the client plans to use the device.

Although the Arduino can detect values between 0 and 1023 on the ANALOG IN 0 pin, the readings were between 40 and 100. This is probably because the Arduino measures voltages between 0 and 5 Volts as the photodiode outputs voltages over a smaller range. Since there are fewer values over which different levels of sunlight may be distributed, the data gathered is less precise.

4.2.3.3 User Input

Based on the desired functionality of the device, the design team decided that two inputs from the user were needed: how long the gate should be open and how long after sunset the gate should close. With these two pieces of information, the timing of all of the functions of the device could be determined.

The design team had many options for user interfaces. The design team chose the interface components based on how easy to find, inexpensive, intuitive to use, and functional they were. The design team decided to use a potentiometer for each of the inputs. A potentiometer is a variable resistor that provides a resistance proportional to the angular displacement of a cylindrical shaft about its centerline.

To test the linearity of their output, the potentiometers were tested with the Arduino. As shown in figure 16, the wiper terminal was connected to ANALOG IN 2, and the other two to GND and +5V (the pin that has 5 Volts of potential relative to GND). They marked various positions of the potentiometer as shown in figure 17, writing the values that the Arduino received at each position next to its mark. The design team determined visually that the values were linear with respect to the angular displacement of the potentiometer shaft. This was important to the design because then a linear function could be applied to the potentiometer values in order to obtain timing values.

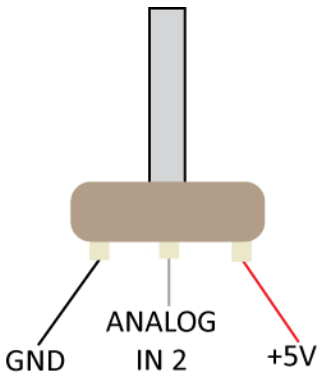


Figure 18: A diagram of a potentiometer and its wiring to the Arduino microcontroller.

Figure 19: Angular displacement markings of the potentiometer shaft with values recorded from the input pin of the Arduino.

The design team decided to attach knobs to the potentiometer shafts such that the user has better leverage on the shafts. The potentiometers had threaded portions in line with the shaft such that the potentiometer could be mounted on a metal plate or plank of wood. When assembled, the prototyped user interface resembled that of a stove. With proper labeling of the potentiometers, the prototype clearly satisfied the objective of "Easy-to-use interface."

4.2.3.4 Programming the Timer

Programming the microcontroller was necessary to process and transmit all of the data flowing in and out the microcontroller. As previously stated, the timing system would require the use of both an internal clock or timer and the external signal from sunset. In addition, the system needs to incorporate the user's inputs: the "period open" (how long the user wants the gate to remain open), and the "delay" (how long the gate should remain open after sunset before finally closing). To write a complete program, all of the aforementioned aspects must be taken into account to send the correct signals to the gate subsystem at appropriate times of day. The team mulled over these different elements of the timing system and came up with a simple, overall structure for the programming code.

The resultant structure revolves around a daily cycle, perpetually recalibrating itself at every sunset. The best way to describe the structure is to follow what occurs throughout the course of a single cycle, as illustrated in the schematic Figure 18. Starting at dusk, as soon as information from the photodiode indicates sunset, the program would initiate a timer. As soon as the timer's reading exceeds the "delay" as dictated by the user input, the gate would close. From then on, the gate must remain closed until it is time to open. To control that, the program would calculate how long the gate must remain closed by subtracting the "period open" from twenty-four hours. The timer would be reset as soon as the gate finishes closing, such that the timer would read how long the gate has been closed at any given moment thereafter. As soon as this time exceeds the calculated amount of time the gate needs to be closed, the program would signal the gate subsystem to open. After this event, the timer simply waits for the photodiode to signal sunset again and repeat the cycle.

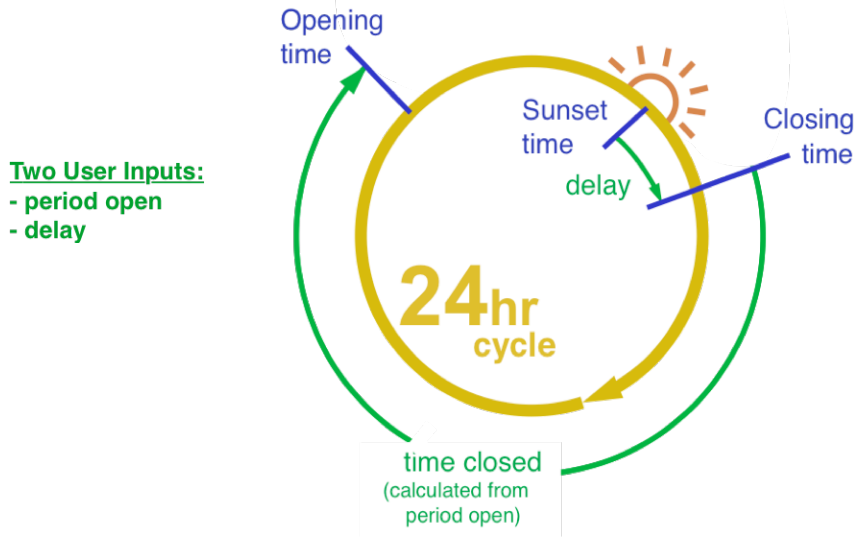


Figure 20: A schematic of the program's cyclical structure. Two user inputs are read from the potentiometers, giving the "period open" and the "delay." Starting at sunset, a timer starts. As soon as its reading exceeds the "delay" input, the gate closes and the timer resets itself. As soon as the timer's reading exceeds the calculated "time closed," the gate opens and remains open for the remainder of the cycle.

The design team chose this particular timing method because of its reliability and ability to self-correct. The team determined that the most important function for the design to fulfill was to close at night, so as to keep the chickens safe from predators like foxes. As a result, it is very important that the device be able to accurately determine when the sun is setting. With this method, sunset is the reference point for the entire timing cycle, assuring that the gate closes at the appropriate time each night. This also fulfills the objective of minimizing the need for user input, since the user could feasibly input the period open and delay times once and never need to worry about the timing again.

4.2.3.5 Safety LED

The team decided that it would be useful for the client to have a safety light to indicate that the gate was open. A small LED can be directly controlled by the Arduino: one lead could be connected to one of the DIGITAL OUT pins on the Arduino and the other lead could be connected to GND. Then, the Arduino could be pro programmed to output a signal on the DIGITAL OUT port when the gate was open. This way, the safety of the chickens in the coop could be checked from far away via the LED.

4.3 Synthesis

As stated above, the design team decided that the vertical lead screw design best fit the client's needs. The final step in the design process was to synthesize the three subsystems into a single functional design. To do this, a few issues regarding the interaction of the power, timing, and gate subsystems needed to be addressed.

4.3.1 Motor Control

The interface between the power, timing, and gate subsystems is at the motor controller. The timing subsystem manipulates the power subsystem via the motor controller to drive the motor. The motor is an integral part of the gate subsystem. It is the translator between electric potential and mechanical work, which makes it the central component of all state changes of the gate.

There are many ways of controlling motors. The design team chose to use a DC motor for its simplicity and common usage, which narrowed the possibilities for motor control. The simplest circuit to control the direction and operation of a DC motor is shown in figure 19.

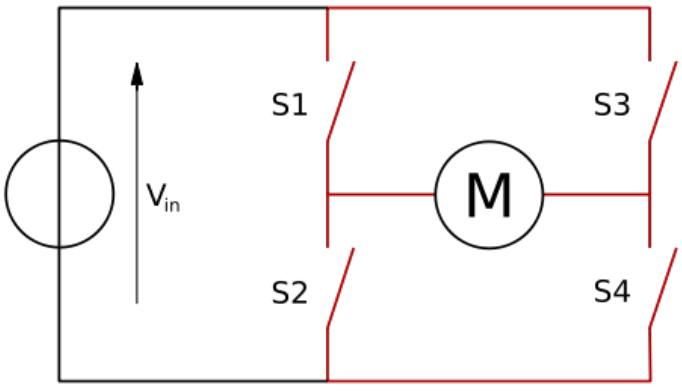


Figure 21: DC Motor controller (Source: www.wikipedia.com)

S1, S2, S3, and S4 are switches that either allow or prevent current flow, and would be controlled by the Arduino microcontroller. Vin is the voltage applied to the circuit, which would be provided by the battery in the power subsystem. When S1 and S4 are connected, then the motor M runs in one

direction. When S2 and S3 are connected, the motor runs in the opposite direction, since the current is applied across the motor in the opposite direction.

The design team found that controlling the motor was a similar problem to controlling the timing. A premade chip called an H-bridge or a circuit made from scratch could be used as the component shown in fig. 19.

H-bridges have the circuit built in, with some extra components for other purposes than what the design requires. The design team found that without technical expertise, the documentation for H-bridges is difficult to follow. The design team decided that an H-bridge would be an unnecessarily complicated way of controlling the motor, and that building a circuit from scratch would be easier for the client to understand, build, and fix.

Building the circuit in fig. 19 from scratch requires a simple electronic component called a relay. A relay is basically a switch that is controlled by another circuit. It has four leads. When a voltage is applied across two of the leads, the other two leads are connected to each other. In the final design, the relays would be used as S1, S2, S3, and S4. The Arduino would activate the right relays at the right times in order to achieve the desired operation of the motor. Thus, the design team felt that in the final design a custom H-bridge circuit using relays would be the most simple and effective way of connecting the power, timing and gate subsystems.

4.3.2 Limit Switches

The design team realized that the Arduino would need feedback about the state of the gate subsystem in order to successfully control the device. Without sensor data about the position of the gate, the system would have no way of reliably stopping the motor once it reached the boundaries of its possible motion. If the motor was stalled when it hit the boundary, even for just ten seconds, it would probably overheat and essentially break. Furthermore, stalled motors draw large amounts of current, which would be a waste of energy accumulated in the battery over days.

The team decided to use limit switches to signal the Arduino when the gate was at its physical limits. The limit switch used, shown in figure 20, connects all three leads when the armature is rotated past a certain angle.

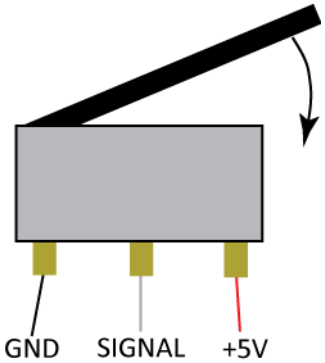


Figure 22: The limit switch

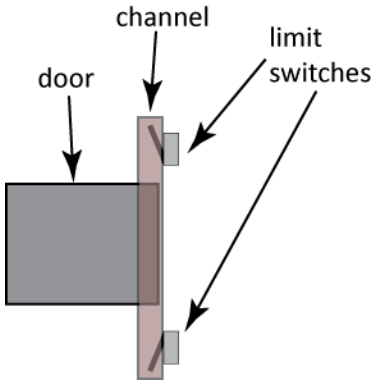


Figure 23: Limit switch placement in final design.

The limit switches used were tested for compatibility with the Arduino. The team connected the GND lead to the GND pin, the SIGNAL lead to DIGITAL IN 2, and the +5V lead to +5V on the Arduino. A pull-up/pull-down resistor was needed between the GND lead and the GND pin or between the +5V lead and the +5V pin such that the circuit would not short when the switch was pressed. With the resistor,

the values read on the DIGITAL IN 2 pin changed significantly when the armature was depressed. Thus, the design team confirmed the ability of the Arduino to reliably take input from a limit switch.

Two limit switches were necessary: one for the upper limit and one for the lower limit of the door's motion. They decided not to use just one limit switch because physically accommodating for the usage of one limit switch would require machining and building precision that is not commonly available. The limit switches were placed as shown in figure 21. The limit switch at the top would activate when the door was at its maximum height. The limit switch on the bottom would activate when the door was at its minimum height.

Note that the bottom limit switch would be perfect for activating the safety LED. When the bottom limit switch is released, the Arduino receives a signal which may be programmed to activate the safety LED.

4.3.3 Microcontroller Programming

The final design of the timing system incorporates the new elements brought in by the selected gate subsystem and the use of limit switches. The program's code retained the same cyclical structure described in section 8.1, with the addition of two different kinds of states. One of them acts as a marker for sunset, while the other determines the status of the gate.

The marker for sunset indicates whether or not sunset just occurred, to help the program discern when to close the gate, as shown in green in Figure 22.

There exist four different states to determine the gate's status: the "opening," "opened," "closing," and "closed" states. Depending on what state the gate is in, the program instructs a different course of actions for any circumstances. When the gate is "opening," the motor is activated to drive the gate upward into an open position. When "open," nothing happens unless a limit switch indicates that the door is no longer fully open for unexpected reasons (i.e. the gate manages to slide down during the day). After the sunset occurs and the appropriate amount of delay time passed, the gate enters a "closing" state, in which the motor is activated to drive the gate downward. As soon as a limit switch indicates that the gate is shut, the state transitions into "closed." In the event that the gate is pushed open, a limit switch would detect this change in position and alter the state back to "closing." From the "closed" state, once the timer reaches the appropriate time, the state switches back to "opening" and the cycle repeats.

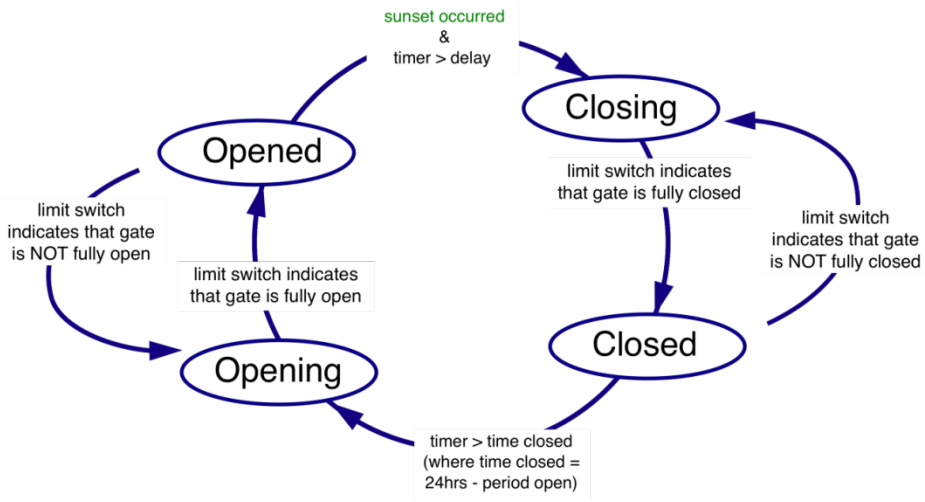


Figure 24. The cyclical process the timing system undergoes each day, navigating between the four different states of the gate with the aid of limit switches, timers

4.3.4 Mounting

All of the components of the gate are attached to one main sheet of plywood in the final design. The plywood sheet can be attached via wood screws or bolts to any surface. In the case of the client's chicken coop, the gate may be attached anywhere on the sides of the coop. It will be necessary to cut a hole in the side of the coop, over which the leadscrew-powered gate will be attached.

4.3.5 Shielding

The final design would also incorporate a shield to protect the leadscrew and the motor. This shield would be constructed out of plastic and would fit around the leadscrew and motor on the side of the gate. Shielding these components would serve the dual purpose of protecting them from environmental factors and preventing chickens and humans from getting too close and possibly getting harmed by the rotating leadscrew.

5. FINAL DESIGN

The final design chosen by the design team consists of a solar-powered gate that moves up and down along a leadscrew, with its motion controlled by an Arduino microcontroller. The leadscrew is powered by a motor, converting the motor's rotational motion into the linear motion of a rider that is attached to the gate. The timing system of the final design uses sunset as a reference point, using readings from a photodiode to determine when the sun has set. An Arduino microcontroller connects all of these components together, recording user inputs through potentiometers, timing the opening and closing of the gate, and processing information from limit switches at the top and bottom of the gate.

Recommendations

The design team came up with several recommendations for improving certain parts of the design. When building the gate-opening subsystem of the design, the team suggests the use of an Acme leadscrew and rider, as these components are specifically intended to bear loads and thus will be more effective than a general threaded rod. The team also suggests the use of a backup door that can be closed over the opening in the coop in case the gate gets stuck in an open position. Prototype testing also revealed that the door is most effectively connected to the leadscrew by a single rider, since using more than one rider can affect the orientation of the leadscrew and make turning it difficult. The team recommends storing the electronic components of the design on a ledge above the actual gate and inside of the coop. This storage area is conveniently located at an accessible height, and can easily be covered by a protective shield to keep the electronic components safe. Finally, it is important to recognize that actual programming of the Arduino will be necessary to make the gate work, and that the motor will need to be interfaced with the Arduino via the H-bridge.

6. CONCLUSION

The design team was tasked with designing an automated gate for a chicken coop. When considering possible solutions for this problem, the team defined guidelines for the actions the design would have to complete, and how well it completed them. The team used these guidelines to divide the task into subsystems, and to create alternatives within each of these subsystems. Evaluation of these alternatives allowed the team to create one final design: a motor-powered vertical leadscrew design with a timing system based off of sunset and controlled by an Arduino microcontroller.

7. REFERENCES

[1] Dym, Clive L., and Patrick Little. *Engineering Design: A Project Based Introduction*. New York, NY: Wiley, 2008.

[2] "Longreach P.O., Australia: Climate, Global Warming, and Daylight Charts and Data." World Climate. 2008. Climate-Charts.com. 29 Mar 2009 <http://www.climate-charts.com/Locations/a/AU94000000360300.php>.

[3] "Ilfracombe - QLD." ExplorOz. 1999 - 2009. I.T. Beyond Pty Ltd. 29 Mar 2009 <http://www.exploroz.com/Places/33905/QLD/Ilfracombe.aspx>.

[4] "14 1/2° Pressure Angle Worm Gears and Worms." McMaster. 18 April 2009. <http://www.mcmaster.com/#57545k527/=1i567m>.

[5] "Coefficients Of Friction." RoyMech Index page. 26 March 2009. 18 April 2009 <http://www.roymech.co.uk/Useful_Tables/Tribology/co_of_frict.htm#>.

[6] "Worm Gears." RoyMech Index page. 23 July 2008. 18 April 2009 <http://www.roymech.co.uk/Useful_Tables/Drive/Worm_Gears.html>.

[7] Norton, Robert. *Machine Design: An Integrated Approach*. 3. New Jersey: Pearson Prentice Hall, 2006.

[8] Rodney Shannon, *private communication*, 2009.

[9] "Frictional Coefficients." RoyMech Index page. 23 July 2008. 04 May 2009 <http://www.roymech.co.uk/Useful_Tables/Tribology/co_of_frict.htm>.

[10] “Microcontroller.” Wikipedia. 2 May 2009. 20 April 2009 <http://en.wikipedia.org/wiki/Microcontroller>.

[11] Mazer, Jeffrey A.. *Solar Cells: An Introduction to Crystalline Photovoltaic Technology*. New York: Springer, 1996.

Appendix 1: Function Means Tree

The function/means tree (in the form of an outline) is not only a way to organize the functions a design needs to perform, it is also a very useful tool to expand a set of basic functions into a larger, more specific set. The design team began with only two main functions that the chook gate would need to perform: let the chickens out of the gate and lock them back in at night. These functions became the highest level functions in the tree--the ones that were the least specific and the least helpful in brainstorming what the final design should do. A functions/means tree works by then breaking these general functions into ways that those functions could be accomplished (means) or into additional 'subset' functions that are a smaller part of a larger task. Then, at each subset function or mean, the process is repeated; means are broken down into a set of functions that need to be accomplished for the specific mean, and functions are either broken into means or divided again into "smaller" functions that encapsulate a more definitive task.

For example, the design team's highest function is to "let the chickens out of the coop." This task can be broken down into five more characteristic steps: (1) activate the opening of the gate, (2) release the gate from its closed position, (3) move it to an open position (so that the chickens can get out), (4) stop the gate at its open position, and finally, (5) resist any force that would close the gate during the time that it is open. These functions, in turn are broken into means that could accomplish them. For instance, "activate opening of gate" could be done by a commercial timer like the ones that turn on sprinkler systems at night, or it could be activated by a "sun-based detection system" that uses a diode or resistor to measure the level of light, current, voltage, etc. Again, for each of these functions and means, there is another level that splits the top items into yet more tasks and ways to go about those tasks. This process continues until the tree reaches a level for each category that can't be broken down any further; in essence, the tree stops when the design team attains the most basic of functions and means.

The tree expanded the team's original list of two functions into a comprehensive series of fifteen. This allowed the design team to see a much larger picture of what a final chook gate design would have to do and played a big role in the creation of the morph chart--the next step in the design process.

Functions are written in black.
Means are written in red.

Let Chickens out of Coop
Activate Opening of Gate
- Timer
  - Collect Energy
    - Solar Panel
    - Extension cord--connect to house power supply
  - Store Energy
    - Battery
    - Capacitor system
  - Keep track of time
  - Accept user input (of changing timer)
  - Store user input
- Sun-based detection system
  - Detect sunrise/sunset
    - "Brightness" detector

Process input from sun-detection system
Monitor current produced in a solar panel
Ammeter connected to panel
Connect timing system to actuating system--sync the two
Release Gate from closed position
Pull a latch open
Motor
Supply power to coop
Obstacle moved from blocking gate
Motor
Collect and Store energy
Battery/capacitor system
Solar Panel
Compress piston that is the latch
Motor
Supply power to coop
Move gate to open position
Let gate swing downhill
gravitational potential stored
Pull open with a spring
spring potential stored
Motor pull open
Supply power
Collect and store energy
Stop gate at open position (lock)
Compress a latch
Run into an obstacle (i.e. a rock)
Spring runs out of potential
Post at open position
Resist closing forces (stay open)
Latch holding in place
Spring potential keeping gate open
Gravitational potential holding gate in downhill position
Obstacle in front of gate (similar to a doorstop)

Lock Chickens into Coop
Activate Closing of Gate
Timer
Collect Energy
Solar Panel
Extension cord--connect to house power supply
Store Energy
Battery
Capacitor system
Keep track of time
Accept user input (of changing timer)
Store user input
Sun-based detection system

Detect sunrise/sunset
"Brightness" detector
Process input from sun-detection system
Monitor current produced in a solar panel
Ammeter connected to panel
Connect timing system to actuating system--sync the two
Release Gate from open position
Pull a latch open
Motor
Supply power to coop
Obstacle moved from blocking gate
Motor
Collect and Store energy
Battery/capacitor system
Solar Panel
Compress piston that is the latch
Motor
Supply power to coop
Move gate to closed position
Let gate swing downhill
gravitational potential stored
Pull closed with a spring
spring potential stored
Motor pull closed
Supply power
Collect and store energy
Stop gate at closed position (lock)
Run into chicken gate frame
Compress latch
Spring compressed all the way
Resist opening forces (stay closed)
Gravitational/Spring potential
Latch, gate bar