VARIABLE SPEED SUMP PUMP

By

Carolyn Petersen

Edward Villasenor

Final Report for ECE 445, Senior Design, Spring 2019

TA: Amr Martini

01 May 2019

Project No. 35

Abstract

Current sump pumps can only run at one speed regardless of how much water is currently in or entering the sump. This paper outlines the design and testing of a variable speed sump pump which would change the pumping speed based the needs of the sump. The speed of the motor that powers the pump is controlled by changing the voltage input to the motor. Voltage control is obtained through a motor controller that uses a PWM signal from a microcontroller to change the voltage based on the duty cycle of the PWM. The PWM signal sent by the microcontroller is based on water level data received from capacitive sensors that are inside the pump.

Table of Contents

1. Introduction

1.1 Purpose

Currently, a one-third horsepower sump pump uses an average of 650 watts per hour [1] and can be prone to flooding unnecessarily. Sump pumps on the market function by having a float switch at the top of a sump which turns a motor on a single speed once the water level in the sump reaches the float switch. As a result, sump pumps often turn on and off loudly during the night. It also creates a flooding scenario where the pump doesn't turn on until the water level is too high in the sump because it can't sense the water rising until it is too late. This projects aims to save energy, detect water rising faster and reduce the loud noise of the pump by making a variable speed sump pump. Instead of turning the engine on and off many times, energy can be saved by varying the speed and running the motor more continuously at a lower speed. There is less noise is this configuration due to the motor running at lower speeds at times instead of only running at high speed. The sump pump would also flood less in situations of high water flow because instead of only turning on once the sump is full, it would detect the high rising water sooner and turn on to its highest setting quicker than a current sump pump could. Building a 250 W sump pump in just one semester is dangerous, so the sump pump that this project is built is smaller than a home sump pump, using only 6 Watts. The end product was a proof of concept, not a ready-to-market device.

1.2 Functionality

The high level requirements for this project that were used in the project proposal were based on the assumption that the motor would use more energy than it ended up using when we rescoped the project. The first high level requirement was that the water intake would increase as the power increased, up to 36 W. While the water intake didn't change, and the entire project ended up using around 6 W, the purpose of the high level requirement was as a way to indicate that water flow rate increased as voltage increased, which did happen. Details on the relationship between power and flow rate are in section 2.2. The second high level requirement was that the sensors could detect water before the water had risen halfway up the sump. This was to test one of the flooding conditions mentioned in the section 1.1 where the variable sump pump could stop certain flooding scenarios by detecting water before it reaches the top of the sump. This requirement was met as the sensors were placed deeper into the sump as seen in the physical design in section 2.1. The third high level requirement was for the pump to pump a reasonable amount of water, which was calculated to be 80 ounces per minute. The 80 ounces per minute was based on the previous assumption mentioned where the project uses 36 W. Since our project used one sixth of that, our flow rate was much lower, at 15.33 ounces per minute, which is about one fifth of the original requirement. The purpose of this requirement was to ensure that the sump pump could actually function as a useful water pump.

1.3 Subsystem Overview

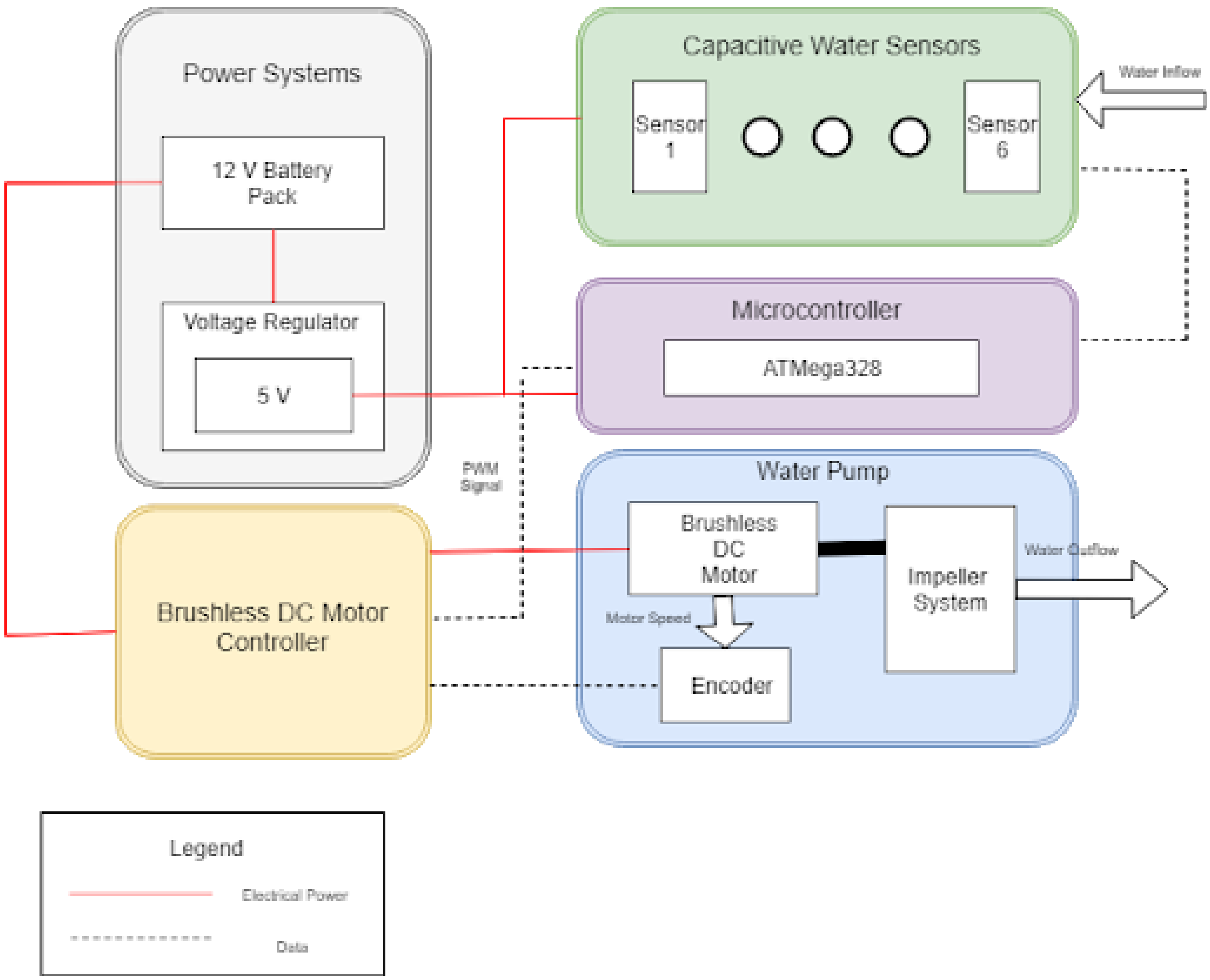

Figure 1: Block Diagram

Figure 1 above is the block diagram for our project. There are 5 major subsystems. They are our power systems, our capacitive water sensors, our motor controller and our water pump.The power systems send 12 V to the motor controller, which sends 5 V to a PCB board with the microcontroller on it. The microcontroller sends a PWM signal to the motor controller, which changes how much voltage the motor controller sends to the water pump. The capacitive water sensors are physically in the sump and send data and get power to and from the PCB board with the microcontroller. The final project did not end up using the encoder as shown on the proposal, as the signal sent from the encoder was noisy and ultimately wasn't necessary due to the linear nature of the voltage control system.

2. Design

2.1 Physical Design

The physical design for this project changed significantly from the proposal. The original design is in Figure 2 below with the table of contents for Figure 2 in Table 1.

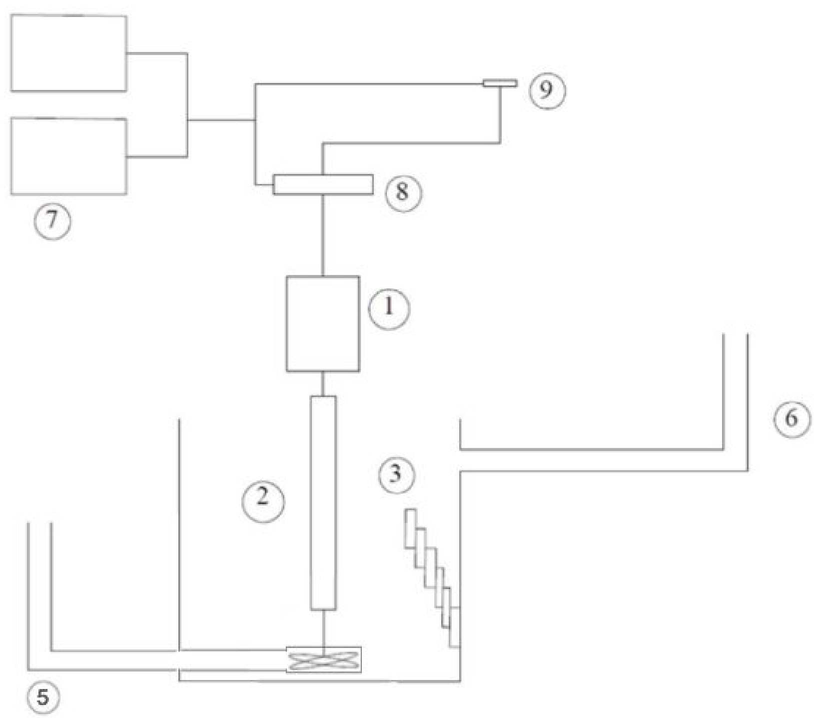

Figure 2: Original Physical Design

Number	**Name**
1	DC motor
2	Axel
3	Sensors
4	Impeller
5	Output Pipe
6	Input Pipe
7	Battery Pack
8	Motor Controller
9	Microcontroller

Table 1: Table of Contents for the Original Physical Design

The major improvement to the design came by making the project a closed system. There are two reservoirs, one which is elevated and the other which has the motor at the top and impeller system on the bottom. The impeller system pumps water from the bottom of the lower water reservoir to the top of the higher reservoir. Water can be put back into the sump with a spigot at the bottom of the higher reservoir. The way the sensors were connected in the sump also had to be modified due to the fact that the sensors were not as waterproof as was assumed in the design document. In order to use the sensors, they were put in a slit at the end of a closed off pipe and sealed. Figure 3 below is a diagram of our final physical design and Table 2 is the table of contents for the new physical design.

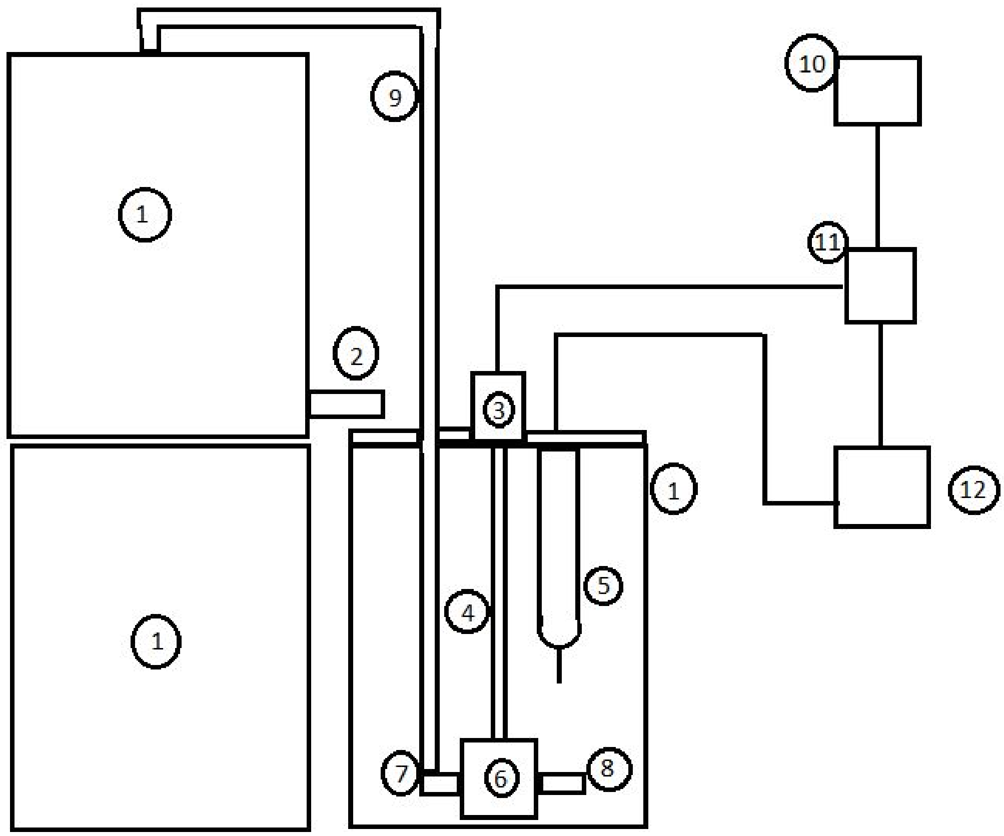

Figure 3: Final Physical Design

Number	Name
1	5 Gallon Bucket
2	Spigot
3	Motor
4	Axel
5	Sensor

Table 2.1: Table of Contents for the Final Physical Design

Number	Name
6	Impeller
7	Output Pipe
8	Intake Pipe
9	Hose
10	Battery Pack
11	Motor Controller
12	PCB Board with Microcontroller

Table 2.2: Table of Contents for the Final Physical Design

In figure 3, only one sensor is shown in the diagram, but in the actual set up there are a total of four sensors in the bucket.

2.2 Power

One of the goals of this project was to save power by running the motor at lower speeds when reasonable. In order to test how much power was used by the water pump at flow rates, the flow rate at each voltage was calculated by timing how long it took the motor to pump water a certain distance of pipe. The volume of water could be calculated from equation 1 below where the diameter of the pipe is .375 inches. L is the length of the pipe that the water moved at the measured time.

$$V = (D/2)^2 * \pi * L \qquad \textbf{1}$$

The flow rate, Q, can then be calculated from equation 2 below. [2]

$$Q = V/t \qquad \textbf{2}$$

At each half volt from the start up voltage at 2.5 V, the flow rate for the water pump was measured. Figure 4 below is the voltage and flow rate data graphed.

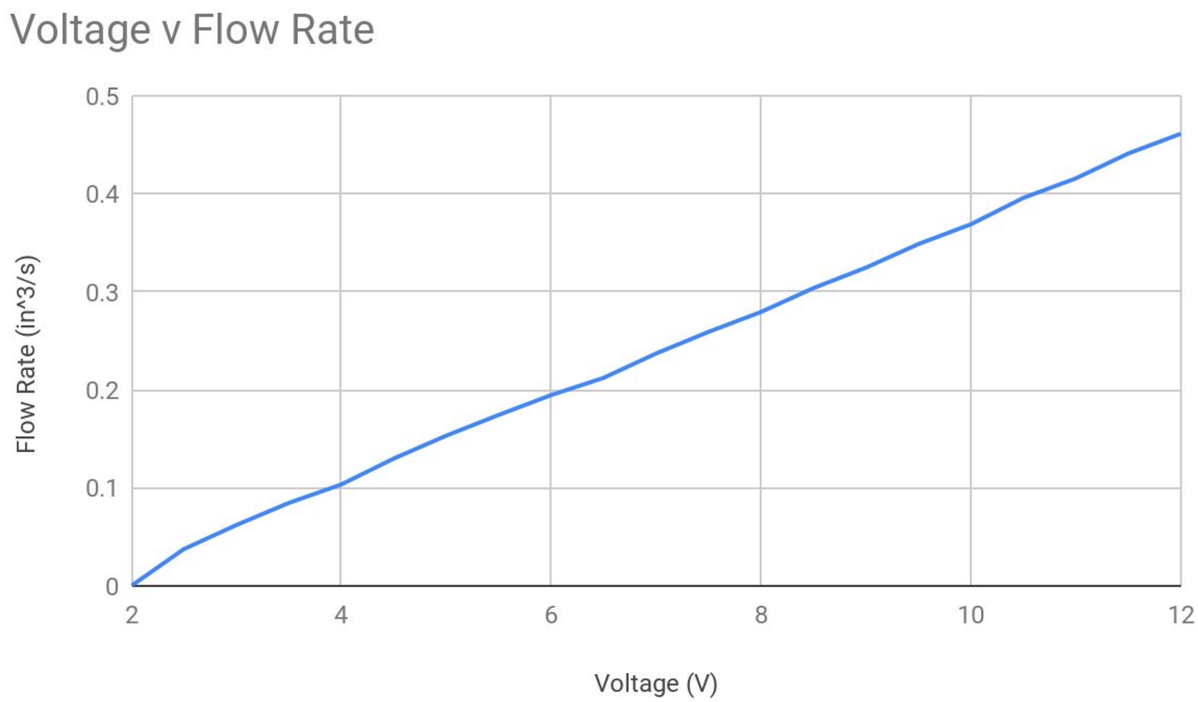

Figure 4: Voltage v Flow Rate

The linear relationship between flow rate and voltage makes it simple to use the voltage control system to find a specific flow rate. This is why, as mentioned in section 1, an encoder was ultimately not necessary as a desired flow rate could be easily obtained. However, in order to find the relationship between power and flow rate, equation 3 below can be used to calculate using the voltage and current.

$$P = V * I \qquad \mathbf{3}$$

In order to measure the current into the motor at each voltage, an oscilloscope was used because the current has a sinusoidal curve. So in order to find the power, the current was averaged at each voltage, and equation 3 was used to create figure 5 below.

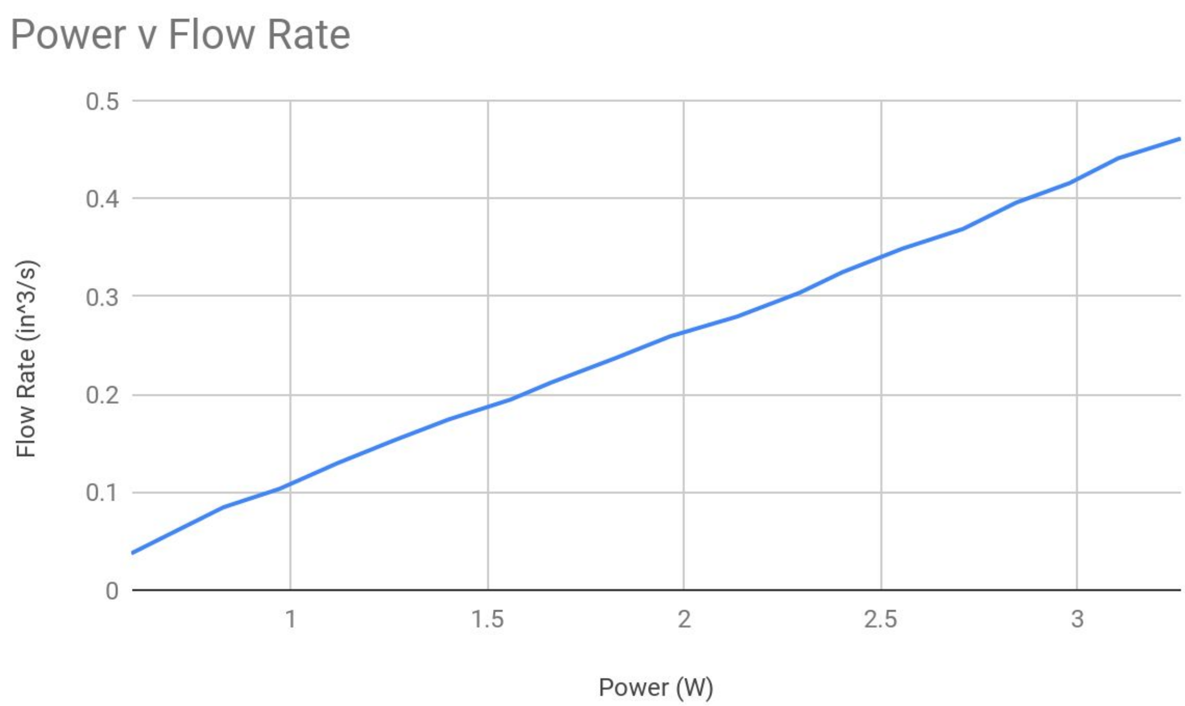

Figure 5: Power v Flow Rate

The relationship in between power and flow rate is also linear. Unfortunately, this indicates that the pump does not save energy. If the pump saved energy, figure 5 would level off horizontally for a range of voltages.

2.3 Water Pump

The water pump is made from a 12 V brushless DC motor whose shaft is connected to an impeller chamber taken from a 1 GPH diaphragm pump. The primary reason why the project does not save energy is likely due to the design of the pump. When the pump is running, the water moves in discrete intervals instead of flowing which indicates that friction is having too strong of an effect on the flowrate to be able to save energy. The motor that is connected to the pump has a maximum of 600 RPM which only translates to .119 GPM. At lower voltages, the motor spins even slower and has a lower pumping rate. This is far lower than the diaphragm pump, so the water pump should have had a motor that rotated significantly faster. The motor in that was originally used in the diaphragm pump spun at 6000 RPM at 12 V which should have been the range that was aimed for when buying the motor to use with this project.

2.4 Capacitive Water Sensors

The way capacitive water level sensors work is that there are at least two “parallel fingers” of a conductive material that has a different dielectric constant than air that is used as the level sensor. You would also have an environmental sensor and a reference sensor. The environmental sensor would represent the capacitance when there is no water near the sensor. The reference sensor represents the capacitance when the water level is completely above a.k.a. covering the sensor. These 3 sensors can be seen in the figure 5 below which shows how the magnetic field interacts with the water level, and picks up a change in the dielectric constant which then corresponds to a change in the capacitance of the sensor.

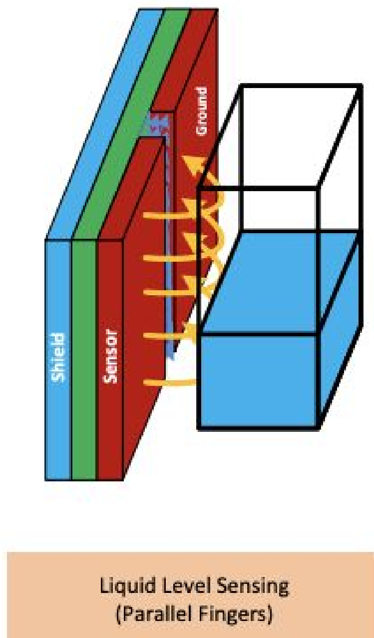

Figure 6: Liquid Level Sensing

2.5 Printed Circuit Board

The PCB (printed circuit board) handles all of the logic for our project. The most significant component of the PCB is the ATmega328P microcontroller which runs software that decides how our project behaves. The microcontroller uses 4 pins to take analog input from the 4 sensors, and uses 1 pin to output a PWM (pulse width modulation) signal to the motor-controller. A secondary function of the microcontroller, is to send the Serial Monitor output back to a computer using the FTDI programmer. This is not necessary though, and is just a feature that eases debugging. The remaining components on the PCB are simple components to support the functioning of the microcontroller. There is a reset button, a pull-up resistor, and a 16 MHz clock among other things.

2.6 Brushless DC Motor Controller

Our brushless DC Motor Controller serves many functions. Its main feature is that it can take a PWM (pulse width modulation) signal and output a voltage between 0 Volts and whatever the maximum regulated voltage is (which should be a little less than the input voltage due to drop out) based on the duty cycle of the PWM signal. But, it can also provide regulated 5 volts to all of the components we need that for, which is the microcontroller and the sensors on the PCB board.

3. Cost & Schedule

3.1 Cost

3.1.1 Parts

Table 3 below contains the part costs for the project.

Part	Manufacturer	Retail Cost ($)	Number	Actual Cost ($)
12 V 600 RPM DC Motor	Uxell	$18.84	1	$18.84
L298N H-bridge Motor Controller	DROK	$7.88	1	$7.88
Diaphragm Pump	Everflo	$33.49	1	$33.49
6V Battery	Rayovac	$5.99	2	$11.98
⅜ inch hose	Eastman	$6.18	1	$6.18
Water droplet depth detector	WINGONEER	$1.60	4	$6.40
ATmega328P	Texas Instruments	$1.50	1	$1.50
Custom PCB	PCBway	$8	10	$80

Table 3: Parts Cost

3.1.2 Labor

The average UIUC EE BS graduate earns $68,000 according to the UIUC ECE website and the average CE BS graduate earns $84,000. [3] Given an average work year of 45 hours/week and 50 weeks/year, the average salary of an EE graduate is $30/hour and the average salary of a CE graduate is $37/hour. When these two are averaged together, the average ECE graduate makes $33.5/hour. The following formula can be used to determine the total labor costs given the length of production time.

$$10\ hours/week\ *\ \$33.5/hour\ *\ 2\ people * 2.5 * 16\ weeks\ =\ \$26,800$$

Parts of the pump were also constructed by the machine shop. The machine shop has an hourly rate of $50 and it took 8 hours to complete. Therefore, our machine shop cost was $400. Added to the salary cost, the total labor costs are $27,200.

3.1.3 Total Costs

The total cost can be calculated by adding the parts cost to the labor cost.

$$Total\ Cost\ =\ Labor\ Cost\ + Parts\ Cost\ = 27,200\ +\ 166.27 =\ \$27366.27$$

3.2 Schedule

Week	Carolyn Petersen	Edward Villaseñor
1/14	Brainstorm Ideas, find teammate	Brainstorm Ideas, find teammate
1/21	Research and familiarize myself with sump pumps	Research and familiarize myself with sump pumps
1/28	Search for parts to be used in our design and weigh the benefits and costs of each	Search for parts to be used in our design and weigh the benefits and costs of each
2/4	Write and submit proposal	Write and submit proposal
2/11	Research how the sensors work. Edit design based on feedback.	Research how the sensors work. Edit design based on feedback.
2/18	Complete Design Document Check. Research how much energy it takes to pump a certain amount of water.	Complete Design Document Check. Research how to program the arduino.
2/25	Complete First Design Review	Complete First Design Review
3/4	Rescope the project and prepare for second design review. Work on a pump design with the machine shop.	Rescope the project and prepare for second design review.
3/11	Research how to pump water. Buy motor and motor controller. Test motor speed relationship with voltage.	Research how to pump water. Buy parts for water-proofing the sensors
3/18	Work on individual progress report	(Spring Break) Unit test the sensors. Start programming the Arduino.
3/25	Test motor controller and arduino with test motor. Test with different duty cycles.	Research, design and construct water-proof sensor apparatus. Order all remaining parts.
4/1	Rough PCB draft in Eagle.	Construct PCB circuit on breadboard. Make revisions to the system with machine shop

Week	Carolyn Petersen	Edward Villaseñor
4/8	Finalize PCB design in eagle and order it through PCBway.	Finalize PCB design and Order it
4/15	Assemble whole system.	Solder and assemble PCB. Assemble whole system.
(4/15 continued)	Test system. Plan how you will Demo. Get and test batteries for a 12 V power source.	Test system. Plan how you will Demo.Integration test the sensors.
4/22	Create final presentation and practice delivery	Create final presentation and practice delivery
4/29	Write final paper	Write final paper

4. Requirements & Verification

4.1 Power System

- Power
    - Two 6 Volt batteries in series were used to power the motor controller, the sensors and the microcontroller. One regulator was used to regulate the voltage sent to the motor and the other was used to send 5 Volts of regulated power to the microcontroller and sensors

Requirements	Verification	Result
Battery must be able to provide 12 Volts of power to the voltage regulator and the motor controller within a ± 10% margin.	1) This can be verified by using a voltmeter to check the voltage of the battery.	COMPLETED. Our two batteries in series provided within ± 10% of 12 Volts at all times to the motor controller.
The motor controller must provide 5 V to the sensors and to the microcontroller within a ± 10% margin.	2) This can be verified by using a voltmeter to check the voltage to the microcontroller and to the sensors.	COMPLETED. The motor controller was used to send regulated power to all components requiring 5 Volts, including the microcontroller and the sensors.
The motor controller should give the motor (1) a maximum voltage of 12 V and (2) a minimum voltage of 2=0 V depending on different PWM signals from the microcontroller.	3) This can be verified by using a voltmeter to check the voltage is 12 V to the motor when the microcontroller sends a signal for maximum voltage and 0 V for minimum voltage.	(1)Incomplete: ~10.4 Volts is the max we could get to the motor because motor controller steps down from 12 Volts, but negligible on performance of pump. (2)Complete.

4.2 Capacitive Water Sensors

Requirements	Verification	Result
Water level sensor needs to be accurate within at least 1.5 inches.	Test the readings sent to the microcontroller against viewing the water level with a ruler and the naked eye.	COMPLETED. Since the height of the exposed pad of each sensor was ~0.75 inches, and the sensor was able to be more accurate than a binary reading, the sensor was accurate by a smaller margin than 0.375 inches.

4.3 Printed Circuit Board (Microcontroller)

Requirements	Verification	Result
Microcontroller is able to send PWM signal to the motor controller.	PWM signals can be checked by measuring the voltage output with an oscilloscope.	COMPLETED.
Microcontroller needs to be able to estimate the water level and rate of change of the water level based on analog signals output by the sensor(s).	a)The analog output can be made human readable through a connection from the microcontroller to the serial com port of a pc. b)This can be compared to a ruler in the bucket.	COMPLETED. But, accuracy could still be improved due to the sensors behaving in a non-linear fashion

4.4 Water Pump

Water Pump
○ Motor: The electrical motor would be an DC motor. Its rated voltage is 12V and rated current is 3 Amps. This is a motor for a RC boat, with a rpm ratio of 3800 rpm/V. This means that at maximum speed can run ideally at 45,600 rpm, but this number is unloaded. As our motor will be pumping water, this means that it will have a considerable load, so its maximum speed would be much lower.
○ Impeller: An impeller is a rotating component of a centrifugal pump which transfers energy from the motor that drives the pump to the fluid being pumped by accelerating the fluid outwards the center of rotation. An impeller is a short cylinder with an open inlet (called eye) to accept incoming fluid, vanes to push

the fluid radially, and a splined, keyed or threaded bore to accept a drive shaft.
○ The motor is connected to the impeller by an axel and the impeller is funneled into an output pipe.

Requirements	Verification	Result
When water reaches the top of the bucket, which will be 1 foot high, the motor should get 12 Volts from the motor controller.	a) This is a DC motor, so the motor runs at its highest speed at its highest rated voltage. The motor that is going to be used for this project is rated at 12 V. [4] b) A voltage probe can be used to check that the voltage to the motor is 12 V when water reaches the top of the bucket.	Partially Complete: ~10.4 Volts instead of our target of 12 Volts but correct 100% duty cycle. Again, having the max motor voltage be 10.4 Volts instead of 12 Volts did not stop the system from completing its high level goals.

5. Conclusion

5.1 Accomplishments

We were able to successfully create a water pump which varied the speed it pumped water at based on the water level inside the pump. So, overall, our project was a success. Unfortunately, we learned that due to our the parts that were chosen for the pump, no energy was being saved. The reason for this was that the speed at which the pump took in water and the friction and pressure it would have to overcome, cancelled out any potential power savings we could have realized by running the motor at a different speed. We also learned a lot along the way. We learned about microcontrollers, motor-controllers, brushless DC motors, PCB design, manufacturing, and assembly, and water pumps.

5.2 Uncertainties

One of the major issues that we ran into was the uncertainty involved in measuring the water level. The problem was that water would stick to the sensors. This would happen when the sensors would first be submerged, then if the water level drops, sometimes some water droplets would stay behind on the sensor, causing it to mistakenly send the signal corresponding to a higher water level than the actual water level was.

5.3 Future Work / Alternatives

In the future, we would use a different sensor to measure the water level. There is a capacitive water level sensor created by Texas Instruments [4] that uses the same technology as our sensors but seems to report a more accurate measurement. The way in which it avoids the setbacks our sensor had is that it never actually comes in contact with water. The way it works is that it is mounted on the outside of the container whose water level we are measuring. The only caveat is that the sides of the container cannot be too thick, otherwise it will interfere with the magnetic field's ability to pick up differences in the dielectric ratio of the sensor to that of when it is fully submerged and that of when it is not near water at all.

We would also use a more powerful motor that spins faster with a bigger water intake system for the impeller so that the water flows instead of stuttering and can save energy.

5.4 Ethical Considerations

As far as our project is concerned, since it closely resembles the sump pumps currently in use around the world, it has essentially the same ethical and safety issues as current sump pumps. Sump pumps are a critical defense for homeowners in protecting their property from water damage. Flooding costs the United States 3 billions dollars a year. [5] Therefore it is critical that the variable speed sump pump is reliable as failure in the pump could costs thousands or hundreds of thousands in dollars to homeowners. Unreliable pumps or incorrectly installed pumps could cause millions in litigation. [6]

In line 1 of the IEEE Code of Ethics, we are to "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment". The way this applies to us is to properly control the safety of the moving parts and to manipulate the water in a safe way. Some safety precautions include things like never allowing children to use it or get too close without adult supervision. Our sump is equally safe as any high-performance home device with a moving part of high rpm (rotations per minute). Caution should be used to ensure that whenever moving parts (impeller, axis, and motor) are exposed or interacted with, particularly for maintenance or replacement, the entire device must be fully powered off and you must wait until all parts stop moving before interacting with it.

References

[1] “Appliance Energy Costs Operating Costs for Major Household Appliances.” *Madison Gas and Electric Company*, 2015, www.mge.com/images/PDF/Brochures/residential/ApplianceEnergyCosts.pdf.

[2] Hall, Nancy. “Mass Flow Rate.” *NASA*, NASA, 2015

[3] ece.illinois.edu, “Salary Averages,” Salary Averages :: ECE ILLINOIS. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages.asp. [Accessed: 21-Feb-2019].

[4] “United States Flood Loss Report – Water Year 2014.” *National Weather Service*, www.nws.noaa.gov/om/water/Flood%20Loss%20Reports/WY14%20Flood%20Loss%20Summary.pdf.

[5] ASHLEY NERBOVIG. “Jury Awards Woman $1.75 Million in Lawsuit against Home Builder, Plumbing Company.” *The Billings Gazette*, Billings Gazette, 21 Aug. 2015, billingsgazette.com/news/state-and-regional/crime-and-courts/jury-awards-woman-million-in-lawsuit-against-home-builder-plumbing/article_9e947ab0-e917-5b11-8006-2c00543515ad.html.

[6] “Capacitive-Based Liquid Level Sensing Sensor Reference Design.” *Texas Instruments*, Mar. 2015, www.ti.com/lit/ug/tidu736a/tidu736a.pdf.

Appendix A Microcontroller Source Code

Main_By_Level.ino:

```
/*
  ||
  ||
```

```
  || MAIN
  ||
  ||
*/

unsigned long prevTime = 1;
volatile unsigned long curTime = 2;

int ledPin = 9;
//int encoderInputPin = A5;

const int sensorArrN = 3;
int sensorArr[] = {0, 0, 0};
int oldSensorArr[] = {0, 0, 0};
float waterLevelRateChange[] = {0.0, 0.0, 0.0};

char buffer[1023];

int motorSpeedN = 10; //MUST BE (sensorArrN*boundsN)+1
//int motorSpeed[] = {0, 120, 144, 165, 180, 210, 230/*, 255*/};
int motorSpeed[] = {
    0,
    120,
    135,
    150,
    165,
    180,
    195,
    210,
    225,
    240
  };


int boundsN = 3;
int bounds[] = {35, 142, 250};//180, 160,

void setup() {
  Serial.begin(9600);
}

void loop() {
```

```
  //  unsigned long timeHigh = pulseIn(encoderInputPin, HIGH);
  //  unsigned long timeLow = pulseIn(encoderInputPin, LOW);
  //  unsigned long encoderFreq = timeHigh + timeLow;
  //  //  float dutyCycle = (float)timeHigh / float(encoderFreq);

  int bufN = 0;
  if (curTime > prevTime) {
    prevTime = curTime;
    //    Serial.println("prevTime");
    //    Serial.println(prevTime);
    //    Serial.println("HIT");
  }
  //  prevTime = curTime;
  curTime = millis();
  oldSensorArr[0] = sensorArr[0];
  oldSensorArr[1] = sensorArr[1];
  oldSensorArr[2] = sensorArr[2];
  sensorArr[0] = analogRead(A0);
  sensorArr[1] = analogRead(A1);
  sensorArr[2] = analogRead(A2);
  int emergencySensor = analogRead(A3);

  waterLevelRateChange[0] = (float)((float)((float)sensorArr[0]) -
((float)oldSensorArr[0])) / ((float)((float)curTime) - ((float)prevTime));
  waterLevelRateChange[1] = (float)((float)((float)sensorArr[1]) -
((float)oldSensorArr[1])) / ((float)((float)curTime) - ((float)prevTime));
  waterLevelRateChange[2] = (float)((float)((float)sensorArr[2]) -
((float)oldSensorArr[2])) / ((float)((float)curTime) - ((float)prevTime));

  int motorLevel = 0; //0 to motorSpeedN-1

  if (emergencySensor < 100) {
    for (int i = 0; i < sensorArrN; i++) {
      for (int j = 0; j < boundsN; j++) {
        if (sensorArr[i] > bounds[j]) motorLevel++;
      }
    }

    analogWrite(ledPin,  motorSpeed[motorLevel]);

    bufN = sprintf(buffer, "sensorArr[0] %d\tsensorArr[1] %d\tsensorArr[2]
```

```
%d\temergencySensor %d\tmotor %d\t", sensorArr[0], sensorArr[1],
sensorArr[2], emergencySensor, motorSpeed[motorLevel]);
    Serial.print(buffer);

    //    bufN = sprintf(buffer, "sensorArr[0] %d\toldSensorArr[0]
%d\tcurTime %d\tprevTime %d\twaterLevelRateChange[0] %f\tmotor %d",
sensorArr[0], oldSensorArr[0], curTime, prevTime, waterLevelRateChange[0],
motorSpeed[motorLevel]);
    ////    Serial.println(buffer);



  } else {
    analogWrite(ledPin,  255);
    bufN = sprintf(buffer, "sensorArr[0] %d\tsensorArr[1] %d\tsensorArr[2]
%d\temergencySensor %d\tmotor %d\t"/*\tencoderFreq %d"*/, sensorArr[0],
sensorArr[1], sensorArr[2], emergencySensor, 255/*, encoderFreq*/);
    Serial.print(buffer);
  }

  Serial.print("waterLevelRateChange[0] ");
  Serial.print(waterLevelRateChange[0]);
  Serial.print("\t");
  Serial.print("waterLevelRateChange[1] ");
  Serial.print(waterLevelRateChange[1]);
  Serial.print("\t");
  Serial.print("waterLevelRateChange[2] ");
  Serial.print(waterLevelRateChange[2]);
  Serial.print("\t");


  Serial.println();
  delay(1);
}




/*
  <200, 200-250,  >250,
  <200, <200,     <200, 200-250,  >250
  0     120       165   210         255
```

```
*/
```