Automatic Control of the System Interface for Wind Turbines

Prepared by:

Matt Baker

Faculty Advisors:

Mr. Scott Rausch

Department Head of Electrical and Computer Engineering

Dr. Thomas Montoya

REU Site Coordinator, Professor of Electrical and Computer Engineering

Dr. Alfred Boysen

Professor, Department of Humanities

Project Information:

National Science Foundation

Research for Undergraduates

Summer 2015

South Dakota School of Mines and Technology

501 E Saint Joseph Street

Rapid City, SD 57701

TABLE OF CONTENTS

Abstract

When operating in extreme wind conditions, wind turbines generate excess amounts of energy. This eventually leads to catastrophic failure, and the turbine becomes inoperable. To combat this, software was created to automatically check wind conditions. When wind velocity hits a critical level, the software shuts down the turbine. Thus, the turbine avoids failure without depending of manual oversight.

1. Introduction

Wind turbines show incredible potential as a powerful, renewable source of electricity. However, wind turbines are only functional at certain wind speeds, and when the wind's velocity is faster than these speeds, turbines generate too much electricity, which can result in fires that permanently damage the turbine. To prevent this, the motors inside the turbine must be designed and programmed to detect dangerous velocities, and shutdown safely to prevent damage to the turbine. This project created the software necessary to defect wind speed and shutdown the turbine blades, both automatically and through human intervention.

2. Broader Impact

Wind energy has proven to be one of the forerunners in the current search for clean, renewable energy. Because of this, it is important to make wind turbines as safe and efficient as possible. This project focuses on creating safe operation for and wind turbine, preventing the destruction of wind turbine when operating at dangerous wind velocities. This will not only make wind turbines safer for people working near them, but will prevent the destruction of thousands of dollars' worth of equipment.

3. Procedure

3.1 Background

Before the beginning of this research, an existing projects was already in place from previous work. This previous project was a Windmill Interface Control (Figure 1) system, created to directly power and control the motor of a wind turbine. The turbine is connected to the interface, and draws its power from it. The interface is responsible for sending commands to the turbine, commands such as “Yaw Break”, “Yaw Motor Left”, “Yaw Motor Right” etc. The interface was previously connected to a physical switchboard, but part of the focus of the project is to program automatic controls in addition to the manual switchboard.

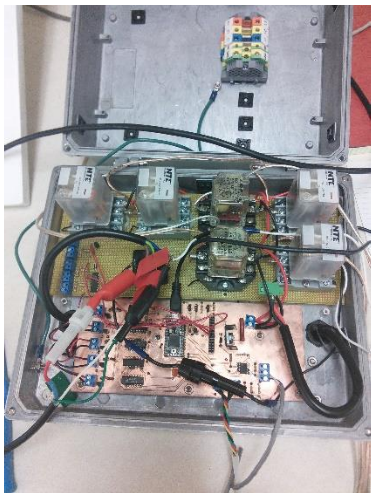
Figure 1

3.2 Hardware

The existing hardware had to be put in working in condition, so that it could power and operate the yaw motor. The power was given by a typical 120 V wall outlet. Then, the motor was connected via the various relays in the control box. Finally, a serial bus was attached to the control box, connecting the pre-existing switchboard to the control box. When power was given to both the switchboard and the control box, a user was able to successfully operate the yaw motor manually.

3.3 Software

The previous project created the necessary software to operate the turbine, so this research focused on creating the program necessary for automatic shutdown. The software was built using teensyduino, an extension of Arduino, the C programming language. Arduino is an extremely useful tool for teal time programming. Arduino operates by running a loop. This means the main section of the program is written, and after it is fully executed, the loop reverts to the beginning and the program runs again. Thus, the code will theoretically run infinitely, barring any failures in operation. By running a loop, Arduino successfully allows for a real-time system check on the physical hardware. So, the implemented wind speed check was created inside the loop, automatically checking to see if critical conditions were meet, and if they were, the emergency shutdown was triggered.

4. Results

4.1 Measurement

The first step to creating the program was recording the wind speed and direction. From the lab testing, an electric anemometer and weather vane were connected to the control box via a USB hub. These instruments digitally recorded the frequency of electric pulses from each instrument and sent the information to a digital input pin in the teensyduino. Then, the raw measurement were converted into useful data: the wind speed converted into miles per hour, the wind direction turning into the degree difference relative to the direction the turbine was facing. The following formulas were used for conversion.

<u>Degree</u>

raw data =ADCval

degree = ((-0.4648)*ADCval + 442.16)

<u>Speed</u>

elapsedMicros waiting

time2 = waiting * 0.000001 * 2

freq = (1/(time2))

Measured_Speed = (0.1009 * (freq) + .0122)

With this information, “degree” represents the direction of the wind relative to the turbine, while “Measured_Speed” represents the speed of the wind. The program now has the data necessary to run the automatic controls.

4.2 Operation

The initial outline for the project was to shut down the turbine when wind exceeded 30 mph. To do this, a function was created in the loop called “AutomaticCheck()”. The function was split into four sub-functions, which ran the testing for emergency shut down. These sub-fuctions are:

- measureDirection();
- measureSpeed();
- findControl();
- sendCommand();

- turnOffMotor();

4.2.1 measureDirection() and measureSpeed()

These functions run the measurement data described previously, so they are vital to the system. By fining the data in the functions, it is available for use in findControl(), where the shutdown check is made.

4.2.2 findControl()

In findControl(), the program will use the speed of the wind to determine how the turbine should react to produce electricity. The function determine what the operate command is chosen to be used in the sendCommand() function. The following chart demonstrates the three categories of wind speed, how the control box will react, and the name of the function which executes the task.

Wind Speed	Production Capabilities	Motor Instructions	Function Executed
0-10 mph	Too slow for electricity generation	1. Turbine Brake On	noOperation()
10-30 mph	Ideal generation conditions	1. Turn Into Wind 2. Turbine Brake Off	normalOperation()
30 + mph	Too fast for safe generation	1. Furl Out Of Wind 2. Turbine Brake On	emergencyShutdown()

4.2.2.1 noOperation()

This is a simple function, it generates the command Turbine_Brake_Engaged.

4.2.2.2 normalOperation()

This function is designed to run while the turbine is generating electricity. In the proper wind conditions, it continually checks which direction the wind is blowing, then determines how the turbine motor should turn so the blades are facing into the wind. To do this, it utilizes findPosition().

The findPosition() function begins by finding the angle between the turbine position and the direction of the wind. (NOTE: The program begins with TurbinePosition equal to zero. So, when the weather vane is installed, ensure that when the measured degree equals zero, the turbine's initial position is also equal to zero. If not, the turbine and the weather vane will not be in sync.) Next, it determine which way the turbine will turn. A negative "TurnDirection" results in turning left, a positive number turns it right. At the same time, the TurnAngle is determined, creating the total displacement needed to be made by the motor. Finally, Turbine position is reset to equal the new wind direction.

After findPosition() is executed, normalOperation() generates the necessary command to turn the turbine, and the Automation is ready to continue to sendCommand().

4.2.2.3 emergencyShutdown()

emergencyShutdown() runs similar to normalOperation, with a few important differences. It also uses findPosition() to determine how the turbine should turn. However, after generating the necessary command, it shifts the displacement of TurnAngle 90 degrees to furl the blades out of the wind. Finally, it sets TurbineBrakeOn equal to 1, so as to lock the brake on after turning.

4.2.3 sendCommand()

With the command properly generated, it is ready to be read by the teensyduino and perform a physical action. sendCommand() determines what direction the turbine should move based on which bits in the command are ‘1’s. The motor turns left if the 15th bit is 1, and it turns right if the 14th bit is 1. So, when findControl() sends the command “0000000000001101”, since the 14th bit is 1, the command is telling the turbine to turn right. This command sets the right motor pin to HIGH, turning on the turbine motor so that it will turn right. In the HIGH state, the motor will turn indefinitely, so the motor must be turned off when the turbine in in its proper position.

sendCommand() works the same regardless of the command generated, and is not affected by wind speed. The total displacement of the motor is decided in turnOffMotor().

4.2.4 turnOffMotor()

This function begins with a delay based on TurnAngle, allowing the motor to turn for the necessary amount of time. The delay is based on how many degrees the turbine needs to turn, multiplied by .1 seconds (.1s is the amount of time necessary for the turbine to turn one degree). Once the turbine has turned the proper amount of time, the motor is ready to be shut off. This occurs in one of two ways. First is using the command Motor_Off. Motor_Off is used to stop the turbine from turning, but it does not turn on the turbine brake, allowing the blades to rotate and generate energy. This command is generated if TurbineBrakeOn does not equal 0. Alternatively, the motor can be shut off using the command Turbine_Brake_Engaged. This command will turn on if TurbineBrakeOn equals 1. This command will activate the turbine brakes, preventing them from generating electricity. Finally, sendCommand() is run again whichever command was generated, shutting off the motor in a safe position, effectively preventing catastrophic failure of the turbine.

5. Discussion

Creating the necessary program to shut down the turbine is largely dependent on the accuracy of the measurement equipment. Unfortunately, the electrical weather vane can provide inaccurate measurements when the vane is less than 15 degrees from the base. Thus, many of turns based on the wind direction cannot be fine-tuned after the initial move out of the wind is made. While this allows for a shutdown which will not cause catastrophic failure, it can cause minor damage to the turbine blades if wind does not hit head on. Improvements should be made to allow proper placement of the turbine in a shutdown mode.

Improvements should also be made concerning the bounds of electricity generation. Currently, the program is designed to run differently depending on three different wind situations: wind velocity under 10 mph, wind velocity between 10 mph and 30 mph, and wind velocity above 30 mph. These values were chosen based on general trends of wind turbines, but should be tested in the future to determine the proper limits of each category, allowing for maximum efficiency.

Finally, the program is currently using a value of 100 for “oneRev”. This means that the turbine will run .1 seconds for every degree it need to turn. However, this value was decided completely arbitrarily, and does not accurately represent the true amount of time needed to turn the turbine one degree. The true value must be found and implemented or else the program cannot work properly.

6. Conclusion

The control box can successfully read data from a weather station and turn the turbine based on the data from the wind. Future work should be done to experimentally find when the turbine should operate at full capacity.

Acknowledgements

The funding for this research came from the National Science Foundation. Thanks for this project to advisor Mr. Scott Rausch, REU site director DR. Thomas Montoya, and to Dr. Alfred Boysen for his assistance and critique in writing and speaking techniques.