DEGREE IN INDUSTRIAL TECHNOLOGY ENGINEERING

SENIOR DESIGN PROJECT

Wind Turbine BreakerBot

Author: Ángel Morenilla Pérez

Director: Miguel Ángel Sanz Bobi

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Wind Turbine BreakerBot

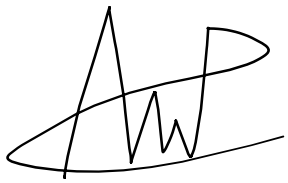
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2021/2022 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.



Fdo.: Ángel Morenilla Pérez Fecha: 25/ 08/ 2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado por SANZ BOBI MIGUEL ANGEL - ***6599** el día 25/08/2022 con un certificado emitido por AC FNMT Usuarios

Fdo.: Miguel Ángel Sanz Bobi Fecha: 25/ 08/ 2022

DEGREE IN INDUSTRIAL TECHNOLOGY ENGINEERING

SENIOR DESIGN PROJECT

Wind Turbine BreakerBot

Author: Ángel Morenilla Pérez

Director: Miguel Ángel Sanz Bobi

Madrid

ACKNOWLEDGMENTS

To Miguel Ángel Sanz for his dedication, patience and effort.

To the members of the I4 team for the teamwork and the joint laboratory hours.

To RWE Renewables for the project proposal.

To Antonio Martín and José Mª Bautista for making the laboratory work possible.

To my family for their concern and support.

WIND TURBINE BREAKERBOT

Autor: Morenilla Pérez, Ángel.
Director: Sanz Bobi, Miguel Ángel.
Entidad Colaboradora: RWE Renewables – University of Texas at Austin – Universidad Pontificia Comillas

RESUMEN

RESUMEN

En este proyecto se presenta el prototipo BreakerBot, destinado para la cabina de control del modelo de aerogenerador GE 1.5 MW SLE, cuya función es detectar el estado actual de los disyuntores y rearmar un disyuntor específico de forma remota. BreakerBot es un sistema de detección, actuación y comunicación con los técnicos, para aumentar el tiempo de funcionamiento de los aerogeneradores.

Palabras clave: aerogenerador, hardware, software, BreakerBot, ensayo, diseño.

1. INTRODUCCIÓN

El crecimiento de las energías renovables como alternativa a los combustibles fósiles se está convirtiendo en una prioridad para muchos países cuyo objetivo es controlar el impacto medioambiental en el planeta Tierra. Así, las energías renovables son clave a la hora de cumplir los objetivos de la "Agenda 2030".

Como uno de los mayores propietarios y operadores de turbinas eólicas de Estados Unidos, RWE Renewables busca constantemente formas de racionalizar el funcionamiento y el mantenimiento de su flota de turbinas. Así, RWE se dio cuenta de que su proceso actual para supervisar y restablecer los disyuntores en el armario eléctrico de la torre superior del modelo de turbina GE 1.5 MW SLE es bastante ineficiente y podría mejorarse.

Con esto en mente, RWE nos encargó la construcción de un prototipo, designado como BreakerBot, de sistema para supervisar el estado de estos disyuntores y poder rearmarlos cuando se disparen.

2. METODOLOGÍA

Así, para conseguir este objetivo los pasos a seguir son:

1. Elección de los elementos de hardware necesarios para cumplir la detección y la actuación del rearme de los disyuntores.
2. Diseño y construcción a medida con la cabina de control de la estructura de BreakerBot.
3. Desarrollo del software de control de los sensores y actuadores del vehículo.
4. Desarrollo del software de la aplicación para controlar el rearme de los disyuntores.
5. Pruebas de BreakerBot en conjunto en el laboratorio.

3. DESCRIPCIÓN GENERAL DEL HARDWARE

En la *Figura 3.1*, que también aparece en la *Sección 6.3*, se muestra la conexión de los elementos de detección y actuación en su diseño final. Estos son:

- Un *Arduino Nano*, que permite la comunicación entre los elementos del hardware y el software.
- Un imán de neodimio, colocado sobre el disyuntor.

- Un sensor modelo *AH49E efecto Hall*, para detectar las variaciones del flujo magnético al cambiar de posición el disyuntor.
- Un actuador lineal modelo *PA-14*, que es el encargado de rearmar el disyuntor disparado.
- Un módulo relé modelo *5V, 10A y 2 canales*, que actúa a modo de driver y controla cuando y hacia dónde el motor debe girar o parar.
- Un motor de paso modelo *12 V DC D8-MOTOR80,* cuya función es mover el actuador lineal horizontalmente a la posición exacta de cada disyuntor.
- Un driver modelo *A4988,* para controlar el motor de paso, indicando la dirección de giro y los pasos.

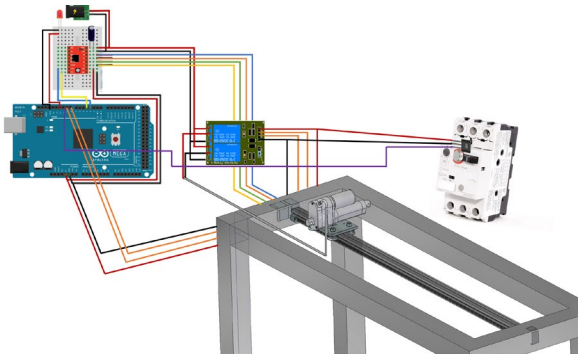


Figura 3.1. Elementos de hardware en BreakerBot

Además de los elementos de la *Figura 3.1*, en el prototipo de BreakerBot también se colocaría un sensor de contacto en la puerta de la cabina para garantizar el correcto cierre.

4. DESCRIPCIÓN GENERAL DEL SOFTWARE

El sistema de comunicación diseñado consta de los elementos mostrados en la *Figura 4.1*.

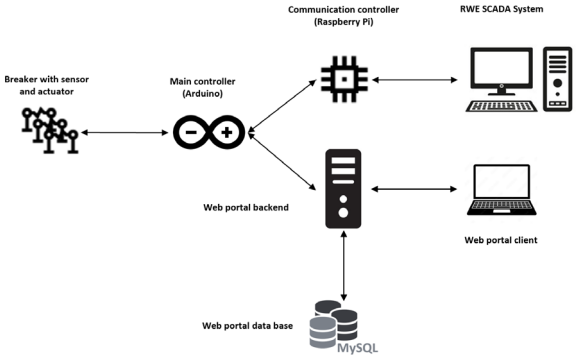


Figura 4.1. Visión general del sistema software

El software diseñado para BreakerBot consta de 3 partes fundamentales: el control, la interfaz gráfica de usuario y la transmisión de datos.

4.1 Control

El control de la detección/actuación del sistema se realiza a través de Arduino IDE, donde se ha desarrollado un código para que los sensores envíen constantes señales al portal web backend que, cuando se dispare un disyuntor y el técnico lo decida, activará el sistema de actuación del rearme. Este sistema de actuación ha sido programado para dirigir el motor de paso a una determinada posición y accionar el actuador lineal, así como para volver a la posición de partida.

4.2 Interfaz gráfica de usuario

El portal web para los técnicos consiste en un backend de MATLAB usando App designer, un frontend para interactuar con el usuario y una base de datos MySQL.

El objetivo de utilizar MATLAB es crear una interfaz gráfica de usuario (GUI) similar a la de un panel de control para que los técnicos puedan tener control sobre el restablecimiento de los interruptores desde la aplicación. Para ello, dentro de MATLAB, se utilizó el App Designer, que es un entorno de desarrollo interactivo para

programar el comportamiento de una aplicación.

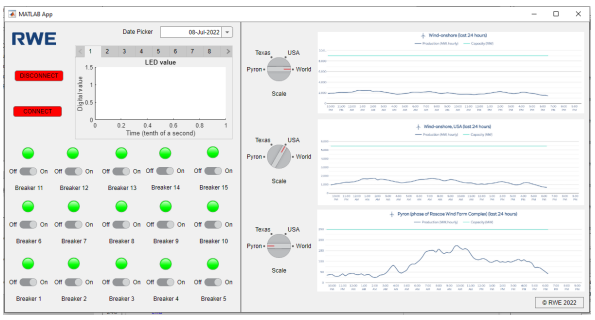


Figura 4.2. Interfaz gráfica de BreakerBot

Además, el backend sirve para recibir datos del Arduino, actualizar los datos en la base de datos, y luego servir estos datos al frontend a través de varios endpoints de la API. Además, aceptará las peticiones del frontend para restablecer un interruptor y las enviará al Arduino para que actúe en consecuencia.

4.3 Transmisión de datos

La otra parte de la comunicación es la transmisión de datos al sistema SCADA de RWE. Así, Arduino se comunica con la Raspberry Pi, que se configurará para actuar como servidor OPC-UA para la transmisión de datos entre sistemas. RWE configurará sus sistemas para recibir esta alimentación OPC. Para obtener los datos de los Arduinos, cada Arduino se conecta a la Raspberry Pi mediante USB.

Una preocupación inicial que surgió al diseñar este sistema fue el cumplimiento de las normas de ciberseguridad del NERC, ya que una solución totalmente personalizada requeriría la gestión del acceso, el cifrado y la certificación del NERC. Sin embargo, debido a que la conectividad se logra a través de la red proporcionada por RWE y el portal web se ejecuta en un servidor ya certificado de RWE, el sistema cumplirá siempre que todos los datos enviados y recibidos se realicen a través de dispositivos en la red de RWE.

5. ENSAYOS DE LABORATORIO

Los múltiples ensayos de laboratorio realizados para alcanzar un modelo óptimo en el diseño del BreakerBot se clasifican en 3 partes principales: detección del disyuntor disparado, actuación sobre el disyuntor disparado e integración y fabricación.

5.1 Detección del disyuntor disparado

Previamente a la elección del sensor del diseño final para la detección de la posición de los interruptores, se realizaron diferentes ensayos, comparando resultados:

- Ensayo del sensor PIR en límites de temperatura de trabajo: propenso a falsos positivos por su sensibilidad a cambios ambientales.
- Ensayo de cámara de visión con reconocimiento: solo reconoce correctamente en espacio luminoso.
- Ensayo del sensor de efecto Hall en límites de temperatura de trabajo: La temperatura tenía pequeños efectos en la entrada analógica que leía Arduino.

Finalmente, se optó por el sensor de efecto Hall cambiando la entrada del sensor de analógica a digital, para garantizar la correcta detección.

5.2 Actuación del disyuntor disparado

El objetivo principal de los ensayos de rearme del disyuntor era asegurar que el motor de paso pudiera colocarse en la posición deseada y que el actuador pudiera producir suficiente fuerza para accionar el disyuntor. Los principales realizados fueron:

- Medición cuánta fuerza se necesita para accionar el interruptor.
- Comprobación de que el actuador puedan ser activados y desactivados desde el microcontrolador.
- Ensayo por separado del actuador lineal y el disyuntor comprobando el rearme y ajustes de los tiempos moviéndose en una dirección.
- Comprobación de que el motor de paso pueda controlarse con el driver.
- Ajuste de la dirección y el número de pasos para alcanzar la posición deseada.

5.3 Integración y fabricación

Los ensayos realizados en la parte de integración e implementación fueron:

- Construcción una réplica 1:1 de madera de la cabina de control encontrada en el nacelle del aerogenerador.
- Construcción inicial del BreakerBot de 3 niveles a madera.
- Construcción final del BreakerBor de aluminio 80/20.
- Montaje conjunto y testeo del BreakerBot en la réplica de la cabina.

Figura 6.1. BreakerBot actuando sobre la réplica de la cabina

6. CONCLUSIONES

La parte fundamental de este proyecto ha sido diseñar, desarrollar la detección-actuación-comunicación y mejorar el prototipo en los múltiples ensayos de BreakerBot. Se ha creado una aplicación que permite a los técnicos comprobar el estado de los disyuntores en cualquier momento y proceder a su rearme cuando lo consideren oportuno, reduciendo los viajes de los técnicos a la torre del aerogenerador y generando mayor beneficio a la empresa al permitir el rearme inmediato a distancia del aerogenerador.

El proyecto cumple la propuesta planteada por RWE y se pone a disposición de la empresa para continuar su desarrollo y mejora. La inclusión de BreakerBot en los aerogeneradores es fácil tanto a nivel económico como técnico, y su integración supondría un gran aumento de la generación de la actualmente sexta granja de aerogeneradores de mayor producción a nivel mundial.

7. BIBLIOGRAFÍA Y REFERENCIAS

[1] Russell Thomas Watford, "Circuit breaker trip notification systems and methods", U.S. Patent 9 054 516 B2, June 9, 2015.

[2] Ronergy – *Top 10 largest wind farms in the world*

[3] Bernhard Jakoby, Isaku Kanno and Loes Segerink, *"Sensors and Actuators A: Physical"*, Elsevier, 2000

[4] DroneBot – *Stepper Motors with Arduino*

[5] Instructables circuits – *DC Motor Controller With Two Relay*

WIND TURBINE BREAKERBOT

Author: Morenilla Pérez, Ángel.
Director: Sanz Bobi, Miguel Ángel.
Collaborating Entity: RWE Renewables – University of Texas at Austin – Universidad Pontificia Comillas

ABSTRACT

OVERVIEW

This project presents the BreakerBot prototype, intended for the control cabin of the GE 1.5 MW SLE wind turbine model, whose function is to detect the current status of the circuit breakers and remotely reset a specific circuit breaker. BreakerBot is a system for detection, actuation and communication with technicians to increase the uptime of wind turbines.

Keywords: wind turbine, hardware, software, BreakerBot, testing, design.

1. INTRODUCTION

The growth of renewable energies as an alternative to fossil fuels is becoming a priority for many countries whose goal is to control the environmental impact on planet Earth. As such, renewable energies are key to meeting the goals of the "2030 Agenda".

As one of the largest owners and operators of wind turbines in the United States, RWE Renewables is constantly looking for ways to streamline the operation and maintenance of its turbine fleet. As such, RWE realized that its current process for monitoring and resetting circuit breakers in the upper tower electrical cabinet of the GE 1.5 MW SLE turbine model is quite inefficient and could be improved.

With this in mind, RWE commissioned us to build a prototype, designated BreakerBot, system to monitor the status of these circuit breakers and be able to reset them when they trip.

2. METHODOLOGY

Thus, to achieve this objective, the steps to follow are:

1. Choose the hardware elements necessary to meet the detection and resetting performance of circuit breakers.
2. Custom design and build with BreakerBot's control cabinet structure.
3. Development of control software for vehicle sensors and actuators.
4. Development of the application software to control the resetting of circuit breakers.
5. BreakerBot testing as a whole in the laboratory.

3. HARDWARE OVERVIEW

Figure 3.1, which also appears in Section 6.3, shows the connection of the sensing and actuation elements in the final design. These are:

- An *Arduino Nano*, which enables communication between the hardware and software elements.
- A neodymium magnet, placed on the circuit breaker.
- A sensor model *AH49E Hall effect*, to detect the variations of the magnetic flux when the circuit breaker changes position.

- A linear actuator model *PA-14*, which is in charge of resetting the tripped circuit breaker.
- A relay module model *5V, 10A and 2 channels*, which acts as a driver and controls when and where the motor should turn or stop.
- A stepper motor model *12 V DC D8-MOTOR80*, whose function is to move the linear actuator horizontally to the exact position of each circuit breaker.
- A driver model *A4988*, to control the stepper motor, indicating the direction of rotation and the steps.

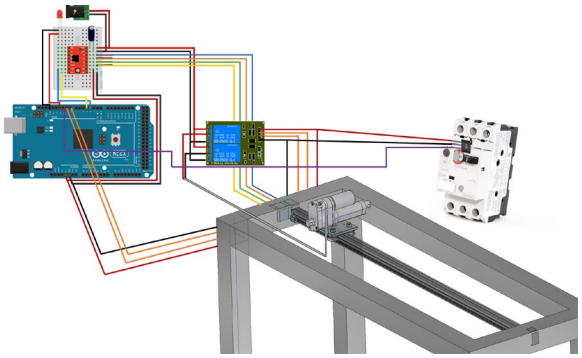


Figure 3.1. Hardware elements in BreakerBot

In addition to the elements in *Figure 3.1*, in the BreakerBot prototype a contact sensor would also be placed on the cabin door to ensure proper closure.

4. SOFTWARE OVERVIEW

The designed communication system consists of the elements shown in *Figure 4.1.*

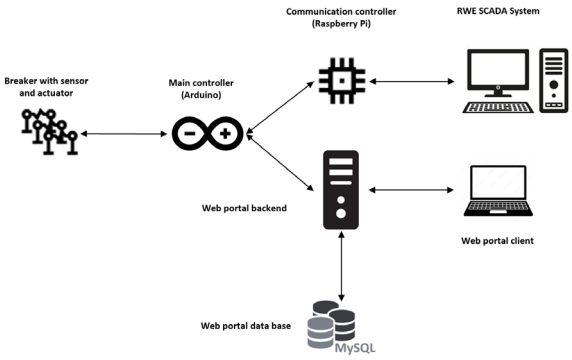


Figure 4.1. Software system overview

The software designed for BreakerBot consists of 3 fundamental parts: the control, the graphical user interface and the data transmission.

4.1 Control

The detection/actuation control of the system is done through Arduino IDE, where a code has been developed for the sensors to send constant signals to the backend web portal that, when a circuit breaker is tripped and the technician decides so, will activate the reset actuation system. This actuation system has been programmed to direct the stepper motor to a certain position and drive the linear actuator, as well as to return to the starting position.

4.2 Graphical user interface

The web portal for the technicians consists of a MATLAB backend using App designer, a frontend to interact with the user and a MySQL database.

The goal of using MATLAB is to create a graphical user interface (GUI) similar to that of a control panel so that technicians can have control over the resetting of the switches from within the application.

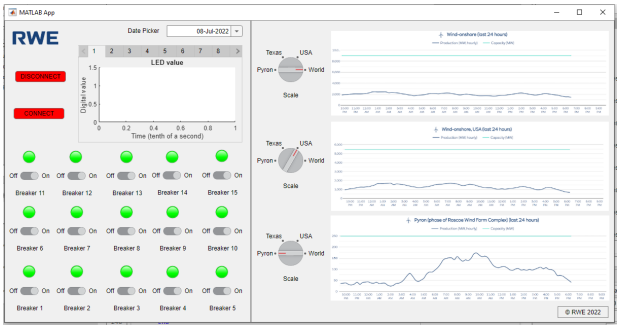


Figure 4.2. BreakerBot graphical interface

To do this, within MATLAB, the App Designer was used, which is an interactive development environment for

programming the behavior of an application.

In addition, the backend serves to receive data from the Arduino, update the data in the database, and then serve this data to the frontend through various API endpoints. In addition, it will accept requests from the frontend to reset a switch and send them to the Arduino to act accordingly.

4.3 Data transmission

The other part of the communication is the data transmission to the RWE SCADA system. So, Arduino communicates with the Raspberry Pi, which will be configured to act as an OPC-UA server for data transmission between systems. RWE will configure its systems to receive this OPC feed. To get the data from the Arduinos, each Arduino is connected to the Raspberry Pi via USB.

An initial concern that arose when designing this system was compliance with NERC cybersecurity standards, as a fully customized solution would require NERC access management, encryption and certification. However, because connectivity is achieved through the RWE-provided network and the web portal runs on an already RWE-certified server, the system will comply as long as all data sent and received is through devices on the RWE network.

5. LABORATORY TESTING

The multiple laboratory tests performed to reach an optimal model in the BreakerBot design are classified into 3 main parts: breaker detection, breaker actuation and integration and fabrication.

5.1 Breaker detection

Prior to the selection of the final design sensor for switch position detection, different tests were carried out, comparing results:

- PIR sensor test in working temperature limits: prone to false positives due to its sensitivity to environmental changes.
- Vision camera test with recognition: only recognizes correctly in a luminous space.
- Testing the Hall effect sensor at working temperature limits: The temperature had small effects on the analog input read by the Arduino.

Finally, the Hall effect sensor was chosen by changing the sensor input from analog to digital, to ensure correct detection.

5.2 Breaker actuation

The main objective of the circuit breaker reset tests was to ensure that the stepper motor could be placed in the desired position and that the actuator could produce sufficient force to operate the circuit breaker. The main ones performed were:

- Measuring how much force is needed to actuate the circuit breaker.
- Checking that the actuator can be turned on and off from the microcontroller.
- Separate testing of the linear actuator and the circuit breaker by checking the reset and timing settings by moving in one direction.
- Checking that the stepper motor can be controlled by the driver.
- Adjustment of the direction and number of steps to reach the desired position.

5.3 Integración y fabricación

The tests performed in the integration and implementation part were:

- Construction a 1:1 wooden replica of the control cabin found in the nacelle of the wind turbine.
- Initial construction of the 3-level BreakerBot in wood.
- Final construction of the BreakerBot in aluminum 80/20.
- Assembly and testing of the BreakerBot on the replica of the control cabin.

Figure 6.1. BreakerBot acting on the replica of the control cabin

6. CONCLUSIONS

The fundamental part of this project has been to design, develop the detection-actuation-communication and improve the prototype in the multiple tests of BreakerBot. An application has been created that allows technicians to check the status of the circuit breakers at any time and reset them when they consider it appropriate, reducing the technicians' trips to the wind turbine tower and generating greater benefits for the company by allowing the immediate remote resetting of the wind turbine.

The project complies with RWE's proposal and is made available to the company for further development and improvement. The inclusion of BreakerBot in the wind turbines is easy both economically and technically, and its integration would mean a great increase in the generation of the currently sixth largest wind turbine farm in the world.

7. BIBLIOGRAPHY AND REFERENCES

[6] Russell Thomas Watford, "Circuit breaker trip notification systems and methods", U.S. Patent 9 054 516 B2, June 9, 2015.

[7] Ronergy – *Top 10 largest wind farms in the world*

[8] Bernhard Jakoby, Isaku Kanno and Loes Segerink, *"Sensors and Actuators A: Physical"*, Elsevier, 2000

[9] DroneBot – *Stepper Motors with Arduino*

[10] Instructables circuits – *DC Motor Controller With Two Relay*

Table of contents

Table of figures

Table of charts

Acronyms

API	Application Programming Interfaces
AVR	Advanced Virtual RISC
DBMS	Database Management Systems
GE	General Electric
GUI	Graphical User Interface
IDE	Integrated Development Environment
IR	Infrared Radiation
LED	Light-Emitting Diode
NERC CIP	North American Electric Reliability Corporation Critical Infrastructure Protection
OEM	Original Equipment Manufacturer
OPC UA	Open Platform Communications United Architecture
PCB	Printed Circuit Board
PIR	Passive Infrared Radiation
RISC	Reduced Instruction Set Computer
SCADA	Supervisory Control and Data Acquisition
SCR	Silicon Controlled Rectifier
SDG	Sustainable Development Goals
SQL	Structured Query Language
UNO	United Nations Organization

Chapter 1. INTRODUCTION

During most of human history, renewable energy has been used to generate heat and power. Only in the last few centuries, fossil and nuclear energy sources were intended for a non-renewable use, creating a strong dependence on them for heating, electricity and transportation nowadays. However, the negative environmental impact and the limited supply of non-renewable energy encourage the use of renewable sources (hydroelectric, wind or solar energies) as a sustainable alternative.

One of the main problems in the field of renewable energies is power generation depending on natural resources that cannot be controlled by humans. Another major problem is the significant economic effort required to install and maintain structures as wind or solar fields. In addition, a failure or breakdown of any component involves a double loss: maintenance or replacement of the failed component and loss of production during the repair time.

As one of the main renewable energy sources, a wind turbine is a device that converts wind force, a renewable energy, into electricity using a simple mechanism: wind blows toward the turbine's rotor blades which spin around and transmit movement directly to the generator or through a shaft and a series of gears that speed up the rotation and allow for a physically smaller generator. The generator converts kinetic energy into electricity which is conducted through the interior of the tower to a transformer that raises the voltage for transport. The electricity generated in each turbine is sent to the substation through underground cables. This is how wind force is converted to electricity in a clean and sustainable manner.

However, wind turbines can fail due to several reasons and be disconnected of the power grid. As wind generators are essential to supply the desired demand, it is necessary to set them back to power generation as quickly as possible.

As one of the largest wind turbine owner and operators in the U.S, RWE Renewables constantly looks for ways to streamline the operation and maintenance of its turbine fleet. RWE recognized that their current process to monitor and reset circuit breakers in the up

tower electrical cabinet of GE 1.5 MW SLE turbine model is quite inefficient and could be improved.

1.1 MOTIVATION

RWE made a proposal to develop a breaker bot which would be installed in the breaker cabinets of their wind turbines to improve the process of detecting a breaker trip and resetting tripped breakers.

Currently, when a circuit breaker trips in the cabinet, RWE receives a notification, but the information provided by the notification does not specify which particular circuit breaker has tripped. This requires a technician to drive to the field, climb up the tower, inspect the breaker cabinet and determine which breaker has been tripped and thus proceed to reset it or do some trouble shooting to fix the issue causing the breaker to trip. This is a very inefficient way of solving this issue since the wind turbine stays off for many hours which is time not used to generate energy and therefore costs RWE a lot of money.

The goal of the design is to allow operators to remotely view the status of each breaker and reset them without being at the tower themselves, providing the operator more insight as to which breaker is tripped and what potential problem is associated with such breaker while at the same time reducing the time it takes for a technician to reset the breaker.

In general, BreakerBot will achieve this by connecting the three following fields: breaker detection, breaker actuation and communication.

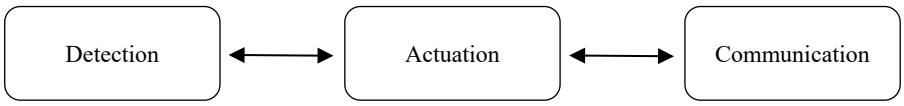


Figure 1. Main fields to be interconnected

1.2 METHODOLOGY

To develop the project, the following tasks are defined:

1. Statement of objectives and delivery of Annex B.
2. Study of the type of wind turbine and the control cabin.

3. Design considerations.
4. Risk reduction.
5. Breaker detection.
6. Breaker actuation.
7. Development of the sensor control software.
8. Development of the actuator control software.
9. Development of the control and communication interface with the technician.
10. Fabrication and integration.
11. Testing and installation.
12. Report writing.

Figure 2 shows a more detailed breakdown of the tasks carried out over time for the development of this project.

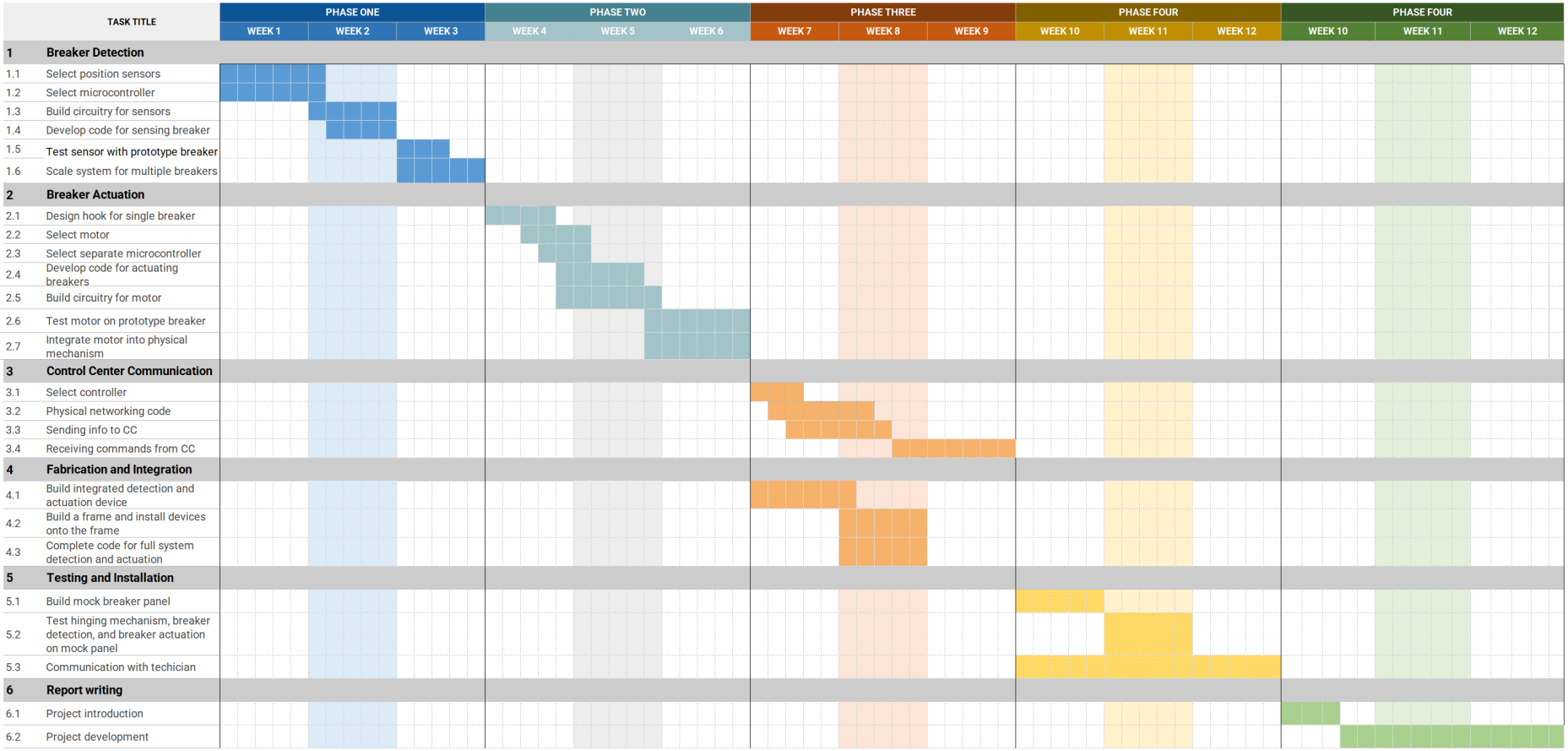


Figure 2. Gantt Chart of BreakerBot development

1.3 PROJECT DESTINATION

1.3.1 RWE

RWE is a German multinational energy company whose electricity generation extends throughout Asia, Europe, and the United States. The company has several energy generation facilities: wind-onshore, wind-offshore, solar, nuclear, hard coal, gas, etc. All its energy sources installations have a total capacity of 43,461.45 MW, hourly, having a net production of 17,904.00 MW. *Figure 3* shows the main sources of electricity generation during 2020 and 2021, where renewables energies were ranked third as their main supply.

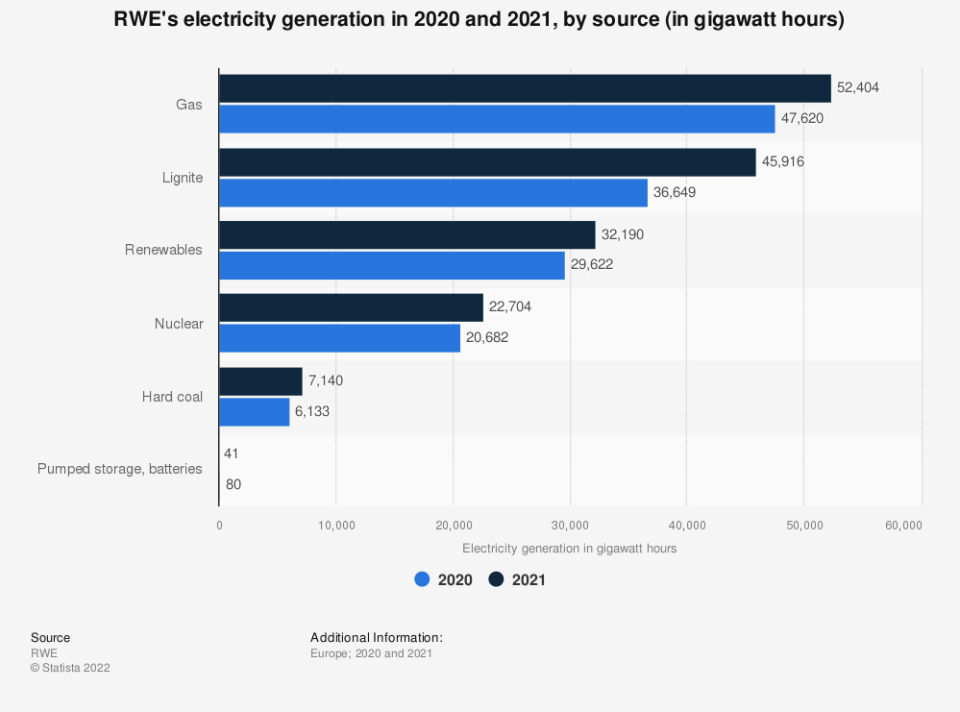


Figure 3. RWE's electricity generation in 2020 and 2021 [29]

Even though RWE's energy generation comes from both renewable and non-renewable sources, the company is one of the international leaders in renewable energy supply. RWE ranks second in offshore wind power generation worldwide and third in Europe's largest producers of renewable energy. Globally, RWE is the fourth largest producer of renewable energy behind Iberdrola SA, NextEra Energy Inc and Enel SpA, as it can be seen in *Figure 4*. RWE is, therefore, in a very comfortable position to capitalize on its position as a leader in renewable power generation in the world.

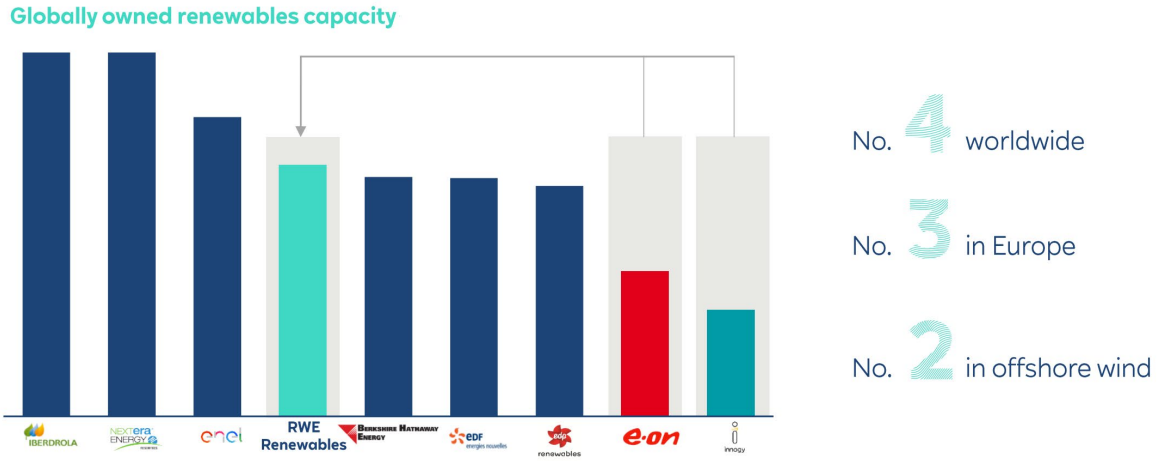


Figure 4. Globally owned renewables capacity [30]

Particularly, RWE Renewables Americas provides maintenance and complete power operations services in the U.S. market. In fact, as shown in *Figure 5*, the company has most of their renewable's facilities in the U.S, where it ranks among the top onshore wind corporations and where the number of their onshore wind projects is growing.

The project proposal from RWE was to improve efficiency in wind-onshore production by diagnosing and repairing faults in some of their older model wind turbines located in the U.S, a strong focus for RWE, where their strategy for renewable energies is growth oriented.

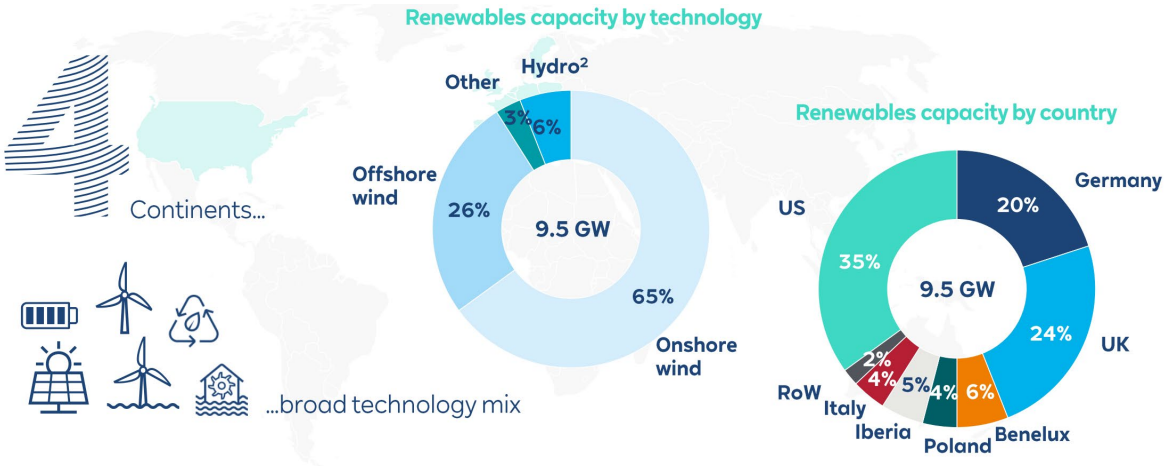


Figure 5. RWE's renewables capacity by technology and country [30]

1.3.2 THE PYRON WIND FARM

Half of the world's largest wind farms are in the United States. As one of the leading states in wind generation in the US, Texas is the main target for the development of wind farm projects. The Pyron Wind Farm, located in central Texas, arose as a result of the Roscoe Wind Farm Project. The Roscoe Wind Farm Project, built of four phases, is classified as one of the largest wind farms in the world. *Figure 3* shows the Roscoe Wind Farm ranks the sixth position in the world ranking based on installed capacity as of 2022 *[24]*.

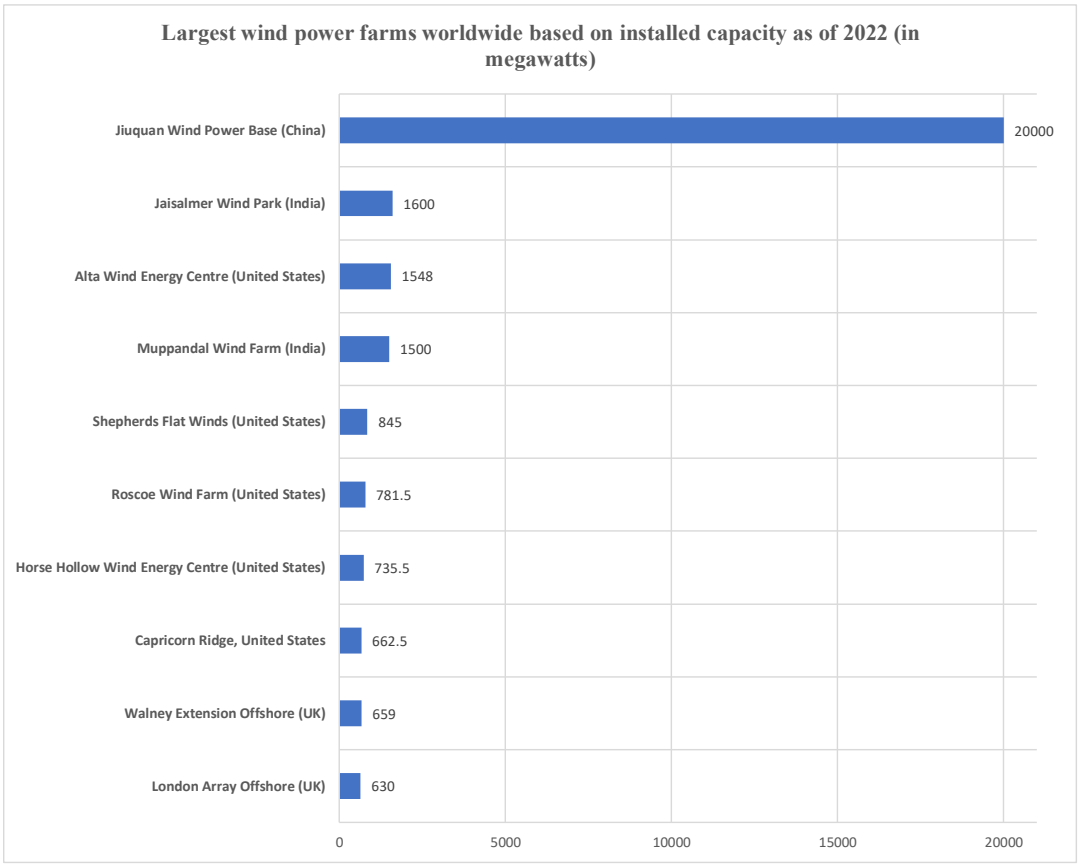


Figure 6. Largest wind farms based on installed capacity as of 2022

This wind farm has a total capacity of 781.5MW of generation, enough to power 265,000 American homes *[23]*. The production is performed by 627 turbines, including 55 Siemens 2.3 MW turbines, 406 Mitsubishi 1 MW turbines and 166 GE 1.5MW turbines. *Table 1* shows Roscoe Wind Farm annual generation, which represents about 0.50% of Texas annual generation *[27]*.

Year	Roscoe	Champion	Pyron	Inadale	Total Annual
	209 MW	126.5 MW	249 MW	197 MW	MW·h
2008	523,383	277,725	-	-	**801,108**
2009	488,402	317,097	506,951	-	**1,312,450**
2010	532,548	349,945	559,509	384,481	**1,826,483**
2011	615,420	416,677	603,565	380,443	**2,016,105**
2012	596,742	413,620	716,988	476,472	**2,203,822**
2013	561,408	393,195	775,605	549,300	**2,279,508**
2014	591,896	406,486	811,344	592,687	**2,402,413**
2015	440,211	349,128	706,057	495,005	**1,990,401**
2016	562,424	308,250	772,988	539,177	**2,182,839**
2017	554,037	366,144	712,817	506,879	**2,139,877**
Average Annual Production (2011–2017)					**2,173,566**

Table 1. Roscoe Wind Farm Generation (MWh)

By using 166 GE 1.5MW turbines, the Pyron Wind Farm can generate 249 MW of clean energy, which represents an emission of million pounds of carbon dioxide if same generation is carried out by using fossil-fuels. *Table 2* and *Figure 7* show the net annual generation of power from 2009-2021 of the Pyron Wind Farm *[25]*. According to these graphs, the average annual generation in the recent past years by the Pyron Wind Farm is 693,447 MWh. As Texas generate about 473,514,913 MWh (in 2020) annually, the project developed in this document has the task to guarantee the 0.15% of the Texas's energy generation takes place, which can be traduced to 63,041 homes being powered.

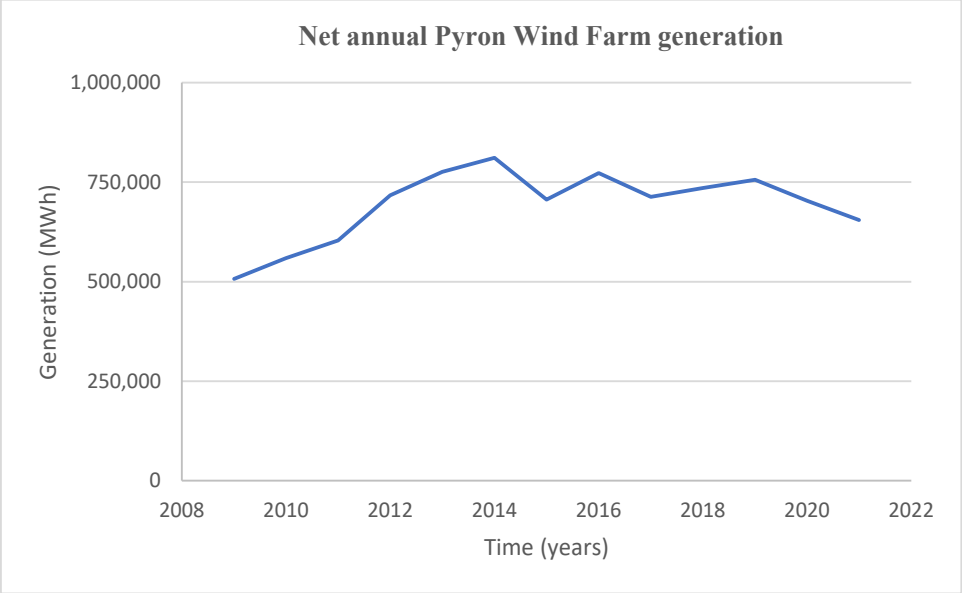


Figure 7. Net annual Pyron Wind Farm generation

Year	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
Generation (MWh)	507,0	559,5	603,6	717,0	775,6	811,3	706,1	773,0	712,8	735,0	756,1	702,9	654,9

Table 2. Net annual Pyron Wind Farm generation

1.3.3 WIND TURBINE MODEL

1.3.3.1 Major parts

As the initial project prototype is designed for a GE 1.5 MW SLE 60H Hz wind turbine, an overview of its features is previously discussed. This wind turbine, composed by three blades, has active control of its yaw, blade pitch and employs a power converter system. In this section, the major components of this wind turbine model, represented in *Figure 9*, are described. The dimensions of this wind turbine are shown in *Figure 8*.

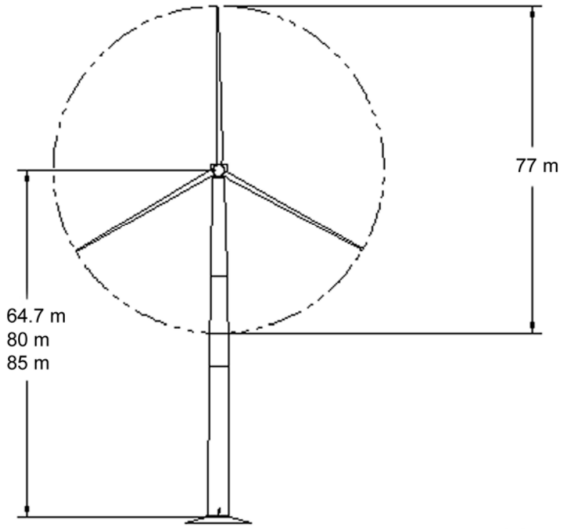


Figure 8. Dimensions of GE 1.5 MW SLE 60H Hz wind turbine

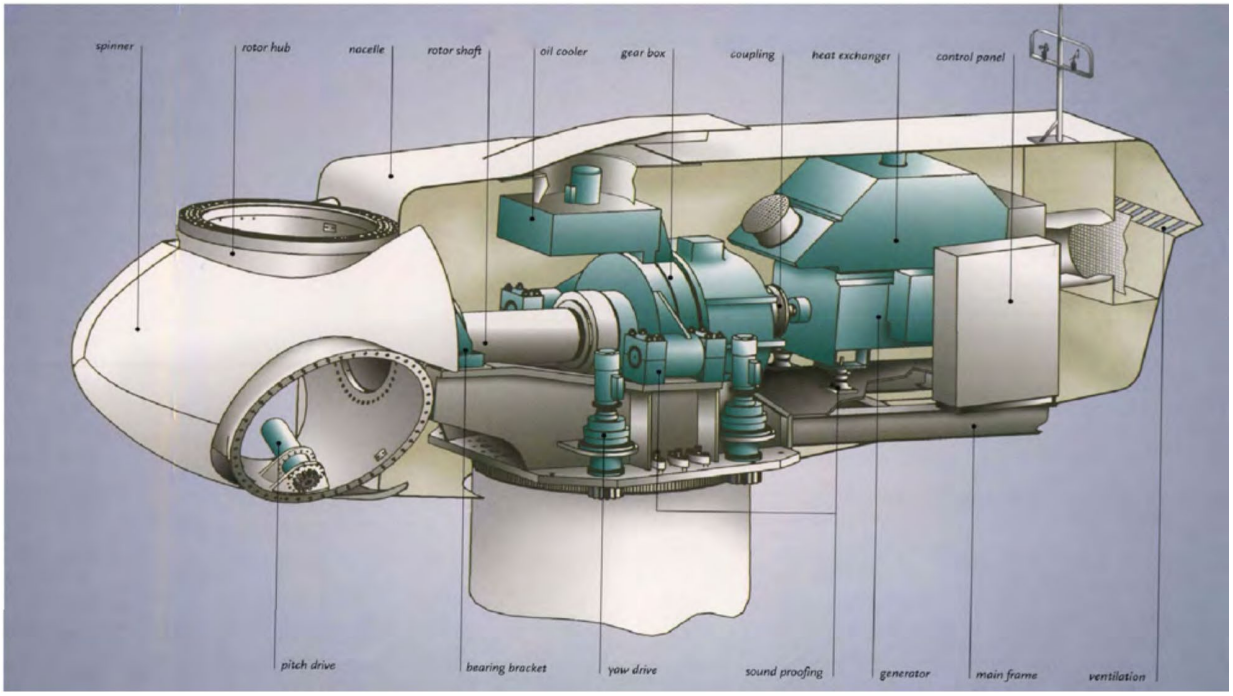


Figure 9. General Electric 1.5 MW SLE 60 Hz wind turbine model [23]

1.3.3.1.1 Rotor

The rotor is the part of the wind turbine that couple the blades to the capsule where the generator is housed. The diameter of the GE 1.5 MW wind turbine is 77 m, sweptwing a total area of 4,656 m^2. This rotor is planned to operate at about 15 revolution per minute. Under normal operating conditions, its rotation is done in a clockwise direction.

1.3.3.1.2 Blades

Blades are design with an aerodynamic design to absorb as much wind energy as possible. The GE 1.5 MW wind turbine model has three rotor blades, which are made of fiberglass epoxy resin, a very resistant but also light material, in order to be able to rotate more easily. Each blade has independent pitch angle control that can change during normal operation. It is essential to give maximum power by adjusting the blade pitch angle according to wind conditions.

1.3.3.1.3 Hub

The hub in the GE 1.5 MW wind turbine model is made of ductile iron and it is used to connect the three rotor blades to the central shaft.

1.3.3.1.4 Gearbox

The function of the gearbox is to increase speed by reducing the torque and transmit power from the turbine rotor to the electric generator. In the GE 1.5 MW wind turbine model, the gearbox has a transmission ratio of 1:72, increasing speed by a factor of 72.

1.3.3.1.5 Brake System

It is used in case the wind speed higher than the one wind turbine can withstand while avoiding failures. There are two main types of braking: aerodynamic and mechanical. Under normal conditions, aerodynamic braking is used, which is achieved by moving the blades away from the wind, preventing the wind from pushing them to rotate. Mechanical braking is generally used as a support for the aerodynamic one, for instance, it is used when the turbine is stopped, such as during maintenance work.

1.3.3.1.6 Generator

A generator converts kinetic energy into electrical energy that is fed into the power grid. A generator consists of a moving part, the rotor, and a static part, the stator. Both parts have windings through which current flows. Currents in the constantly rotating rotor winding induce electromagnetic fields and, therefore, electric current in the stator windings. In the GE 1.5 MW wind turbine model, the generator synchronous speed is 1200 revolution per

minute (rpm), and its rotor has a variable frequency converter to allow the generator to operate at speeds from 880 rpm to 1600 rpm. When working at 1.5 MW power output, nominal speed is 1445 rpm.

1.3.3.1.7 Yaw System

The yaw system is responsible to rotate, depending on the wind speed and direction, the wind turbine around the tower axis and determine the angle of the blades. Therefore, along the aforementioned power electronics, the aim is to find the optimum speed and position to take maximum advantage of the wind energy.

1.3.3.1.8 Anemometer and Wind Vane

An anemometer and a wind vane along other devices, as a lightning rod, are assembled on the top of the nacelle to capture information for the yaw system. The anemometer's function is to measure wind speed, while the function of the wind vane is to measure wind direction.

1.3.3.1.9 Nacelle

In the GE 1.5 MW wind turbine model, the nacelle is made of fiberglass and coated with sound-insulating foam to reduce acoustic emissions. The function of the nacelle is to cover all the parts that make up the wind turbine to avoid its deterioration.

1.3.3.1.10 Tower

Its function is to raise the nacelle high enough to allow blades to rotate and to achieve desirable wind speeds. The GE 1.5 MW wind turbine tower sets the rotor at a height of 65 m, 80 m o 85 m in order to approach different configurations. It is made of steel to support all the weight and it is usually hollow to allow access.

1.3.3.1.11 Electrical Configuration

Figure 10 shows the electrical configuration of the GE 1.5 MW wind turbine generator. The configuration has a power converter on the rotor side, and it is connected to the grid through a power inverter.

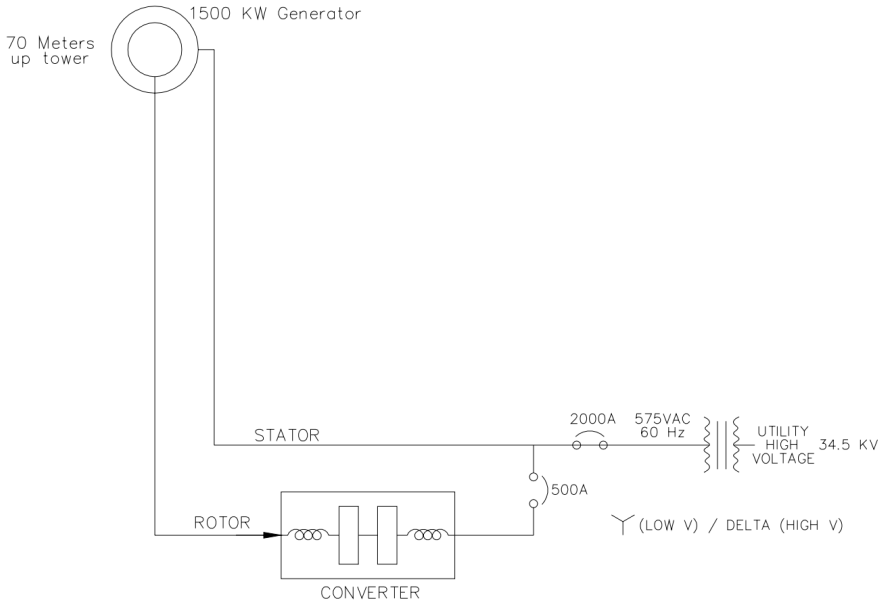


Figure 10. Electrical configuration of GE 1.5 MW wind turbine generator [22]

Wind power can be predicted by using the wind power curve, which relates the wind power to the wind speed. *Figure 11* represents the power output (MW) given by GE 1.5 MW model depending on wind speed (m/s).

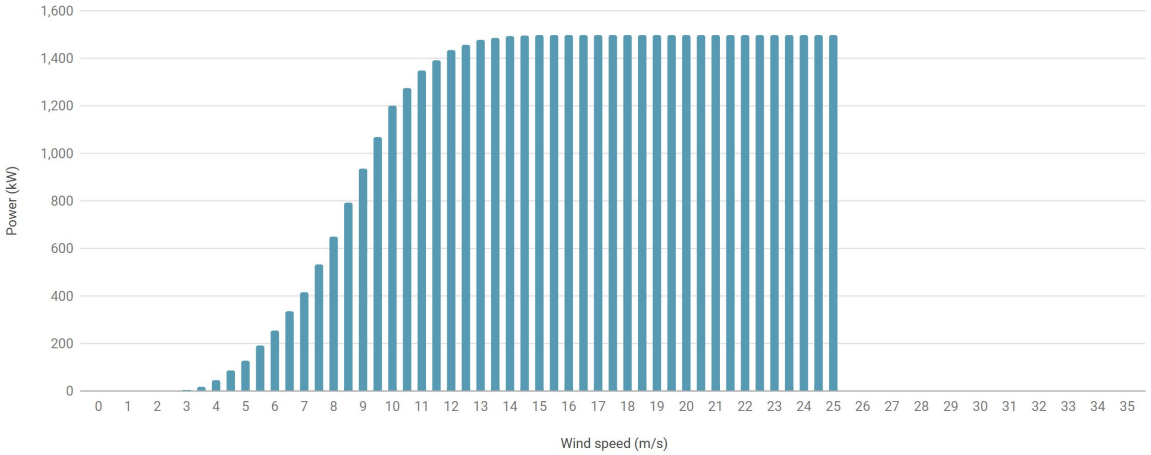


Figure 11. Wind power curve of GE 1.5 MW wind turbine model [22]

1.3.3.2 Specifications for the design

In addition to the previous considerations on the GE 1.5 MW wind turbine model, to be taken into account in the final design, the project has some limitations that restrict it in terms of size, material to be made of, actuating devices to be used and communication.

System Quality	Requirement
Size of solution	122cm H x 58.5cm W X 38cm D
Operating Temperature	-20°C - 40°C
Cybersecurity	NERC CIP Standards
Breaker Actuation Torque	18lb in
Modularity	Fully removable by technician

Table 3. Specifications for the design

Chapter 2. DESIGN CONSIDERATIONS

The objective of this project is to design, implement, test and provide a clear and comprehensive report of the final status of BreakerBot. BreakerBot is a remotely-controlled appliance capable of detecting and rearming tripped breakers inside a breaker cabinet in the nacelle of wind turbines. The task in mind is with building a prototype system to monitor the status of these circuit breakers, present the current status of the breakers, and attempt to reset a specified breaker remotely on command. This document will collect the precise process followed to approach a device that is efficient, economical and scalable to be deployed across multiple turbine types. Starting by detailing the design problem, the design solution, and how it was implemented. Then, testing procedures and protocols will be demonstrated, as well as results. Finally, time and cost considerations as well as safety and ethical aspects of the solution will be discussed.

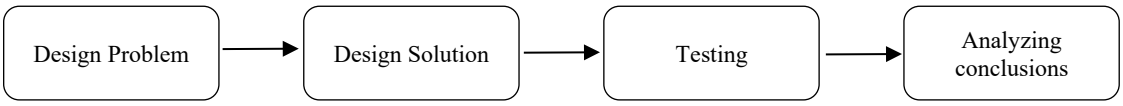


Figure 12. General project's process overview

2.1 DESIGN PROBLEM

Given RWE's current inefficiency when identifying and fixing a tripped circuit breaker in the uptower electrical cabinet of their wind turbine model GE 1.5 MW, the task proposed by the company was to create a solution to help optimize their process by allowing technicians and operators to view current circuit breaker statuses and attempt to reset them remotely. RWE's current process once a circuit breaker has tripped, outlined below in *Figure 13*, requires that a technician be sent to the tower in order to climb up to the cabinet and reset the breaker.

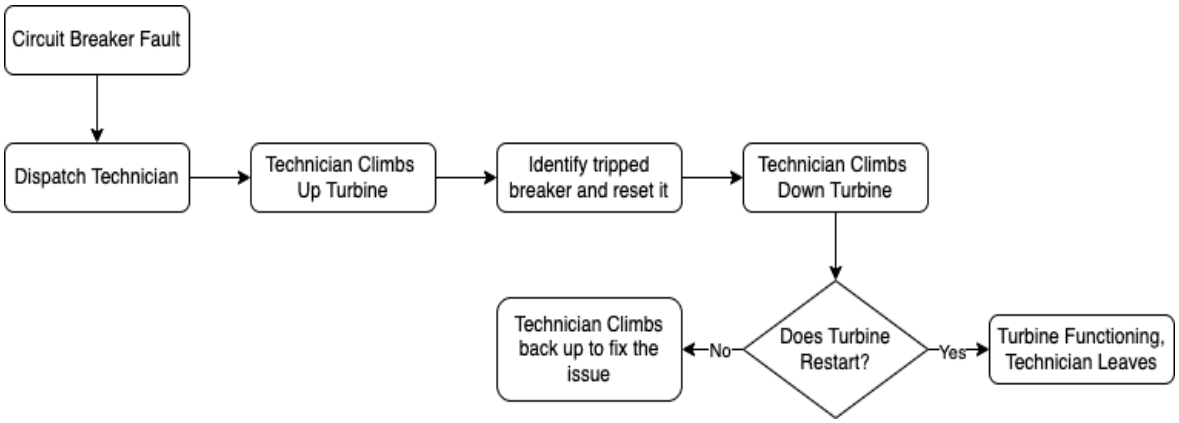


Figure 13. Current process to fix faulted circuit breaker

Once reset, the technician attempts to restart the turbine, and should the turbine not restart the technician must climb back up the tower to diagnose and fix the issue affecting the turbine. Prior to climbing the tower, the first time the technician has no indication of which breaker has tripped, and thus cannot begin to isolate or prepare for any required service. With this in mind, RWE raises the following question: since the actual process of identifying which breaker has tripped and resetting a breaker is simple once you are at the uptower cabinet, can a system be built in the cabinet that allows technicians and operators to remotely view the current status of individual circuit breakers and attempt to reset them before going to the turbine?

With this question in mind, the challenge was to build a system that can monitor the status of individual circuit breakers, relay this information to technicians and operators, and reset circuit breakers after the command of a technician/operator. This system would help optimize the process RWE follows to fix a faulted breaker by allowing technicians or operators to try and reset the breaker remotely. This new process, outlined in *Figure 14*, would reduce the trips technicians take both to the tower and uptower, as should resetting the breaker allow the turbine to restart a technician does not have to go to the turbine at all, and should the turbine not restart the technician will know which breaker was originally faulted and can prepare to work on that part of the turbine.

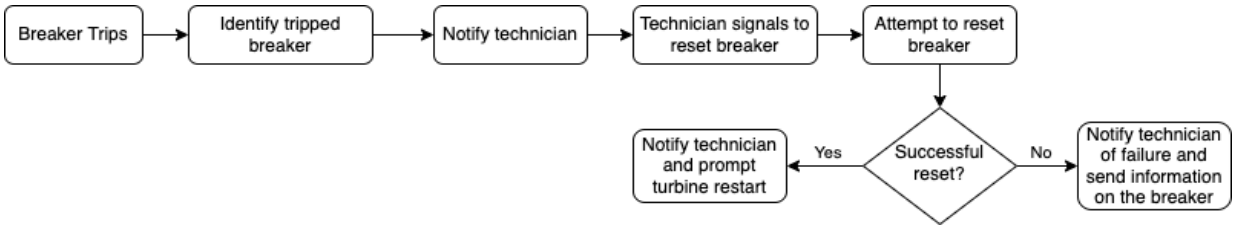


Figure 14. Desired Process to Fix Faulted Circuit Breaker

For the created system to be of most value to RWE, it should integrate with RWE's existing data monitoring systems as much as feasible. Further, as this system would be installed inside the uptower electrical cabinet, it must not greatly hinder access inside the electrical cabinet nor compromise the cabinet's seal from external elements (dust, water, etc.). Beyond these requirements, the system is subject to specifications defined by the tower it operates in, which is the model GE 1.5 MW SLE, presented in *Section 1.3.3.*

2.2 STATE OF ART

RWE Renewables asked for a design of a system that can identify flipped circuit breakers in their wind turbines' electrical cabinets, attempt to reset breakers, and report any tripped breakers to their operations center. The solution needs to fits into the small electrical cabinet and allows technicians to work in the cabinet unhindered, by either remaining relatively small in footprint or by being easily removable.

As there is no prior art on the problem specifically that would not involve changing the entire breaker system, as automatic reset switches, the research on prior art and patent information was focused on three components: moving the system horizontally to a breaker, physically resetting the breaker, and detecting a breaker flip. The articles and prior research in these areas will help guide the design and compare my initial thoughts to what is currently accepted/utilized, while the patent exploration allows me to see if there are any existing solutions that can be used in those specific components.

2.2.1 PRIOR ART EXCLUSIVE OF PATENT INFORMATION

Firstly, the market offers a product called 'SwitchBot Bot' *[2]* that can be stuck onto light switches, and it will automatically switch it from on to off. The device has only a one

direction press, and it can be programmed to switch the interrupter at a certain time and set a certain press-and-hold duration of seconds. Many different models can be found online that include different sensors and utilities. The main idea of this design is very similar to the project to be developed: to automatically switch buttons when required. Instead of being stuck to one simple breaker, the device will have to detect the location of the breaker that it has to flip and move there to reset it. Moreover, the SwitchBot Bot can be connected to Alexa, Siri or any other intelligent assistants. In the case of this project, it will have to be connected to the company's server to store the Data and inform them about any dangerous situations. Furthermore, the technique employed in this device is only pressing or turning a breaker. BreakerBot will have to consider the option of having to flip a breaker. Seeing the simple functionality of this device and the small size, some of its features may be useful for the project.

The initial design entails a component to reset a breaker attached to another component that can move horizontally to the specific breaker. To facilitate this linear movement, two potential design solutions were found, a telescopic arm and rail guided movement. There were some recent examples of telescopic arm design in engineering publications, as that solution would be much more technically challenging.

The main focus of the research was an article that covers the design of a bendable telescopic arm meant to access chaotic or small areas *[3]*. While the bending motion is quite interesting, Much more interesting was the linear extension and contraction of the arm. The engineers sought an arm that could be extended piece by piece, instead of the traditional design which extended all pieces equally. To accomplish this, they use a square shaft to push each piece of the node by a pushing plate, which is shaped so that the piece only moves when that plate has been rotated. This could be extremely useful to access specific breakers, as the pieces could be designed to be the same length as the distance between the breakers' switches. Thus, being able to just push the number of pieces necessary to reach that breaker instead of having to keep track of the current arm's length. However, the contraction of the arm is dependent on the ropes used for bending, which would force to occupy more space in the tower's cabinet and add additional complexity.

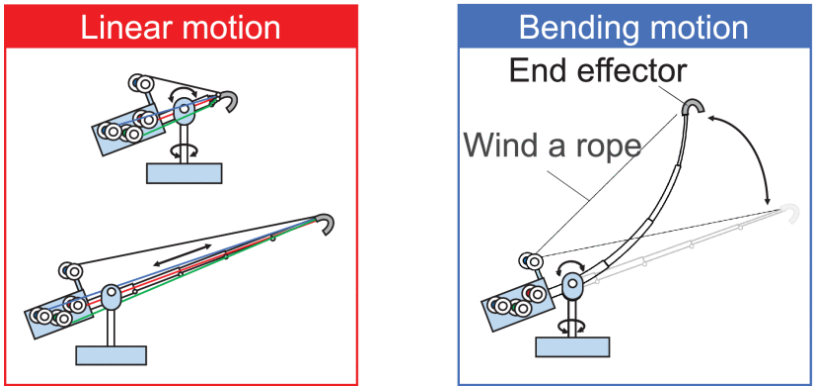


Figure 15. Concept of tendon-driven elastic telescopic arm [3]

In addition to the aforementioned components, a component that can monitor breaker positions and identify which breaker has flipped is required. Two initial design solutions are determined to accomplish this, using physical sensors or video processing software, both of which have plenty of research to explore.

Firstly, current research on sensors to monitor breaker position was done, which led to an article outlining a piezo electromagnetic monitor that is able to stick onto a breaker *[4]*. The prototype leverages a piezoelectric conductor placed between two magnets to monitor the current through the breaker via the magnetic field created by the wire inside the breaker. While this research was prompted by the growing need to know live electrical usage, it may be possible to leverage this for the solution by monitoring whether the current through the switch reaches zero.

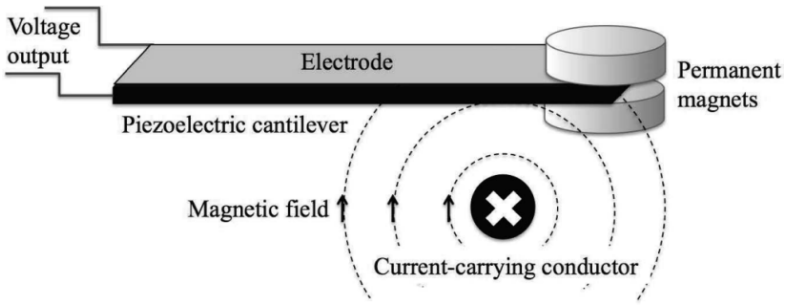


Figure 16. Schematic diagram of a PEM current sensor [4]

An article exploring how to apply video recognition technology to substation monitoring was also found, including monitoring the position of circuit breakers *[5]*. This article from 2011 outlines how video recognition could be used to monitor different equipment inside a substation, which isn't particularly useful for the project, but does cover and show the success in monitor breaker position with video recognition. The article doesn't go into particular depth on the algorithm used to classify if a breaker is flipped but does explain the general steps taken to process the video feed and create an image that can be used to identify a flipped breaker. More than anything, this shows that video recognition software is a viable option and further research into those algorithms could be worth exploring.

2.2.2 PATENT SEARCH AND FINDINGS

The three previously identified components of the design, movement to the flipped breaker, physical resetting of a breaker, and monitoring/detecting which breaker has flipped, which respectively have plenty of commercial applications and corresponding patents. These patents provide existing solutions to specific challenges in the project so, in order to better understand the existing solutions, patent searches were compartmentalized to investigate each component individually.

2.2.2.1 Movement to breaker

Given the space constraints of the wind turbine electrical cabinet and the concerns over technician's ability to perform work in the cabinet, solutions that allow to move the apparatus to the flipped breaker with a relatively small footprint are being sought. One avenue researched is that of telescoping arms, so that when the design is not in use it maintains the smallest footprint possible. The primary patent researched for this method of movement is US patent number 5,410,944, Telescoping Robot Arm With Spherical Joints *[6]*. This patent describes a robotic arm that is much more complex than mine will need to be, with wrist joints at both the base and end of the arm to provide a much larger range of motion, however the primary benefit of this patent is the telescoping section. The arm consists of a central and intermediate piston housed in a final casing forming a three section telescoping arm. Each piece is splined so that there is no rotational movement, ensuring the arm is limited to one degree of movement freedom. The arm is hydraulically actuated, which will likely be too

complex for my solution, however this arm design provides a good baseline for designing the movement system.

Another method of movement is to have a wheel mechanism along a rod that spans the cabinet. US patent 9,972,981 B2, Device For Servicing Live Power Lines [7], accomplishes something similar with movement along power lines. This patent uses a traveling wheel mechanism with accompanying clamping jaw to move along power lines. Wheel driven movement along a stationary track is relatively intuitive to design, however with the other forces that are present in the intended design (namely resetting the breakers) it will be necessary to ensure that once the system is in position that it does not move or rotate. The clamping jaw outlined in this patent is a great solution to that challenge.

2.2.2.2 Resetting Breaker

In researching patents regarding this component of the project the most comparable commercial application found is just a traditional switch (*e.g.* light switch), since while the physical dimensions of the breaker may differ the design problem is almost identical. US patent 7,608,793 B2, assigned to Black & Decker, outlines the design for a remote-controlled wall switch actuator *[8].* This patent uses a yoke and applicable linkage to flip a traditional light switch on/off after receiving a signal from an external remote. How this is designed will be useful in how the actuator is designed if the system chosen covers the breaker. This design also allows for the manual use of the light switch by including sensors that detect the position of the switch, which is especially applicable to my project as that is an additional component of the design problem.

2.2.2.3 Detecting Breaker Flip

Two primary ways to detect a breaker's position have been identified, using sensors to determine the physical location of the breaker (e.g., infrared sensors, proximity sensors) or using a camera with accompanying image processing software. Most of my patent research for this component of our project focused on using sensors, as most software applications/algorithms are not patentable and the few computer-based solutions that are patented describe the system itself as opposed to the actual algorithm/code.

The Black and Decker *[8]* patent discussed above (US 7,608,793 B2) provides one method of detecting breakers have flipped. By using the yoke to move the switch, the system is also able to monitor the position and location of the switch using some form of position sensor. This provides cohesive integration between the component to reset the breaker and the component to monitor the breaker.

US patent 9,054,516 outlines a system for monitoring a circuit breaker panel and the subsequent notification systems *[9]*. This patent is quite broad in both methods of detection and the software systems to notify/record the tripped breaker. It covers detection through a beam sensor that simply identifies if a breaker has tripped, a beam sensor with reflector that can detect the distance and in turn identify the specific breaker that has tripped, a position sensor on each breaker, as well as transmitters within the breaker that can identify the status of the breaker and send that status to the notification system. It covers notification systems that transmit updates by LAN/HAN which triggers a notification to be sent to a broad range of devices, local facilities to trigger a phone call over a telephone line, notifications via a mobile application, local WiFi, and the Internet Cloud. Ultimately this patent doesn't provide much specific information for the design considerations, however it shows a potential roadblock in the solution's patentability since this patent covers a large portion of the problem description.

2.2.3 IMPACT OF PRIOR ART SEARCH ON DESIGN DECISION MAKING

Overall, by combining most of the techniques shown, it is possible to get a very good idea of how to develop the project and implement most of its features.

The prior art regarding the horizontal movement of the system shows that a telescopic arm will require considerably more design work and diligence than thought. The biggest benefit of a telescopic arm was always its form factor, but the clamping design found in the patent search makes a rail-guided movement system much more viable. A clamping design would allow the system to easily be removed from the rail by a technician, alleviating the concern

that it would take up too much space. Ultimately, this research showed that a rail-guided movement system can work and will be much simpler to design, create, and maintain.

The research on detecting tripped circuit breakers has shown that there are many more options than initially expected, and this component will require more thought and consideration than most of the system. The piezo electromagnetic sensors would be simple to create and easy to install, but would require the most research. The physical sensors would provide the most concrete information but could cause the most issues regarding space. Using video recognition software would minimize our footprint but will require a massive amount of time to implement.

However, the most interesting and likely most impactful research was in resetting the breaker, as the information uncovered by the Black & Decker patent shows that these components don't have to be as separate as they were treated. The yoke design presented in that patent can give the status of a specific breaker, thus the system to reset a breaker and detect a tripped breaker would be combined. If an adapted version of that design were used and one were installed on each circuit breaker, that would completely eliminate having to switch to a circuit breaker, simplifying the overall design. The largest consideration at that point would be the simplicity of removing the solution so a technician could work in the cabinet.

2.3 DESIGN SOLUTION

The solution can be divided into four main subsystems: breaker detection, communication, breaker actuation, and system integration. *Figure 17* below shows a block diagram of the solution. Everything starts at the breaker. When any magnetic sensor detects a trip, it sends a signal to the Arduino, which sends information to two different sources. First, Arduino gets relayed to the Raspberry Pi, which connects to RWE's datapoint monitoring SCADA system where operators may monitor it. Then, Arduino processes and sends the data updates to the technician portal frontend. From this web portal, technicians are able to send a reset signal that travels back up the communication chain to Arduino, who will then activate the corresponding linear actuator, actuating the tripped breaker.

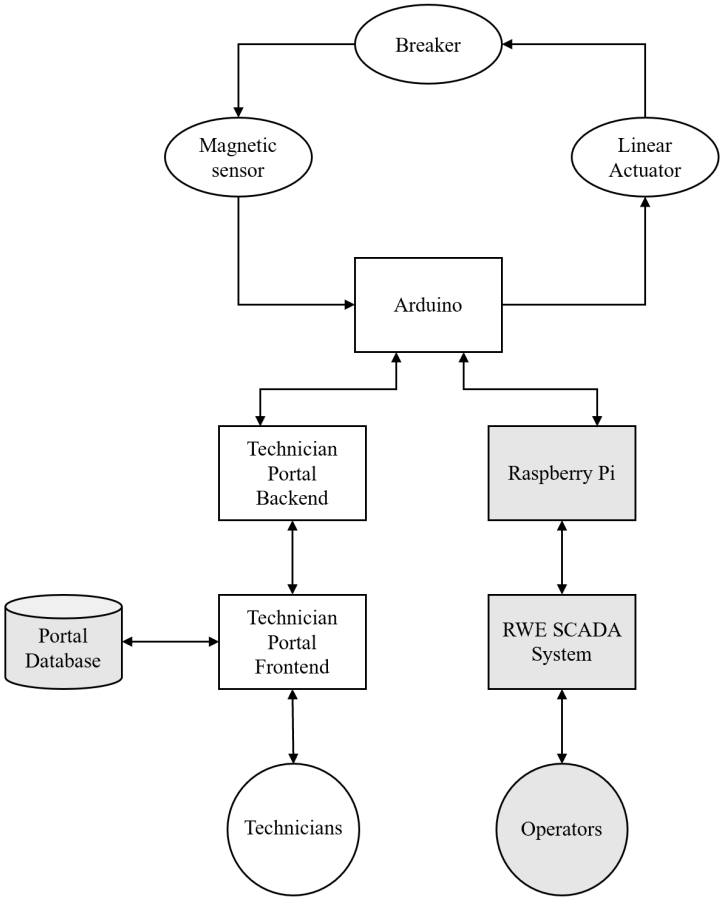


Figure 17. High-level overview of BreakerBot

As it can be seen in *Figure 17*, BreakerBot's objective is to perform non-colored functions, while auxiliary color functions must be implemented by RWE to connect the project to their existing system.

2.4 RISK REDUCTION

As part of the design process, some primary risks associated with the project were investigated and studied to mitigate them. The results of these investigations can be seen in both the integration (panel design) and general control systems (cabinet and panel sensors). The findings of the risk reduction work helped guide future decisions and were important to approach final design considerations. The eight risks investigated are:

- Technician access

- Proper reinstallation of BreakerBot
- Operation at extreme temperatures
- Fragility and durability
- Technician safety and automatic shutoff
- Overvoltage
- Power supply and power loss
- Proper breaker detection

2.4.1 TECHNICIAN ACCESS

A primary concern in the design was ensuring that technicians would still have full access to the breaker panel to perform necessary repairs and any design that hinders that would ultimately cause more delays than the time saved. This research had the task of achieving a design that provided the greatest level of access for both, a design that covers all the breakers and a design that moves linearly to a breaker. Conclusions led to a hinged design that allows the device to move away from the panel and gives the technician full access to the breakers. Another option would be a design that fits snugly into the cabinet (122cm H x 58.5cm W x 38cm D) without leaving any space, so that the design is compact inside the cabinet and does not suffer any deflection when vibrations or other external factors occur. The latter design would be easily removed when the cabinet is opened.

2.4.2 CORRECT REINSTALLATION

If by chance a technician must remove BreakerBot, it is necessary to make sure that the device has a way of knowing it is correctly put back into place. This is extremely important to avoid potential hazards and reduce the risk of damages as a result of misalignments.

To achieve this 100% security of having placed BreakerBot correctly back into place, an alternative would be using touch sensors with LED lights (*Figure 18*) or a laser with a white panel (*Figure 19*).

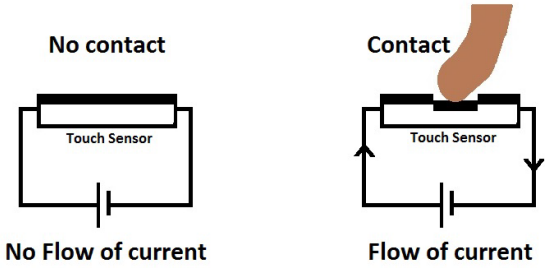


Figure 18. Touch sensor diagram [19]

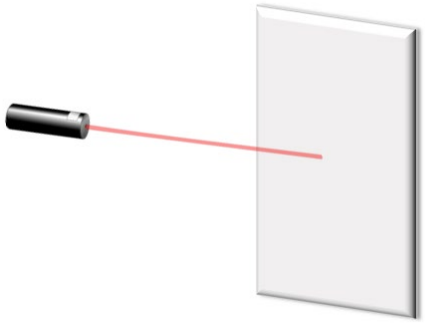


Figure 19. IR sensor diagram

The touch sensor will be stuck onto a corner of the junction between BreakerBot and the panel. If the Bot is correctly placed the touch sensor will be conducting current and letting a LED light shine. Consequently, it can be known if the apparatus is correctly put into place.

The idea of the laser consists in the following: if BreakerBot is well-placed the laser light will go all the way through, and the laser light will be detected on the white panel.

After some experimenting and thoughts, the conclusion reached was that the LED lights were a better option than the laser since it is easier to install, and it can be easily and quickly detected which breaker is causing the problem. Moreover, the laser is much more difficult to properly install because if it is misplaced only by a mm or it accidentally moves from its spot due to vibration it will be useless.

2.4.3 OPERATING TEMPERATURE

As the wind turbines, the system is designed to have an operating range of -20°C to 40°C, it is necessary to ensure that the system will operate in that range with no performance decline particularly at the extremes. These temperatures would mainly impact three aspects of the problem posed: controllers used, breaker actuation, and movement to a breaker. Most microcontrollers that could be used for the solution (i.e., the Arduino UNO R3) have operating temperature ranges that greatly exceed that of the tower. Similar results were reached with breaker actuation looking at servos/motors that might take part of the final solution, finding that while not all motors fall into this operating range there are plenty of industrial grade options to use. Ultimately, while confirming that the motors and controllers selected have acceptable operating temperature ranges, it shouldn't impact the design. Investigating movement to a breaker (primarily for actuation), two approaches of linear movement were considered: stepper motor and leadscrew, and a telescopic arm/piston.

The findings for the stepper motor were that its mechanical part is mainly made of stainless steel, which has a high thermal resistance that would allow the mechanism to behave the same within the temperature operating range of the turbine model of this project.

To investigate a telescopic arm, a small experiment was done: an aluminum telescopic arm was tested at different temperatures close to operating range limits to check its mechanical behavior. It was first tested its performance/movement at room temperature (20°C) as the baseline. To test its performance at the lower temperature extremes, the telescopic arm was placed in a freezer (approximately -18°C) for two hours before again testing its performance. To test the higher temperature extremes, the telescopic arm was placed in an oven set at approximately 75°C for 45 minutes, reaching a temperature over 40°C.

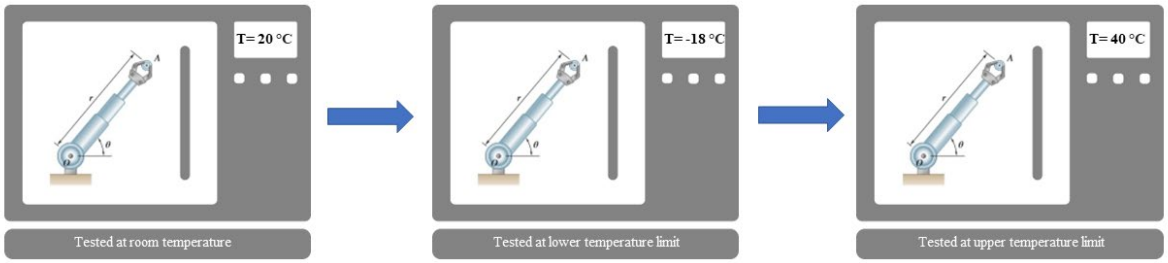


Figure 20. Testing aluminum telescopic arm at boundary temperature limits

Then the results of this simple experiment were compared. The performance of the arm slightly increased at the higher temperatures, with the arm being easier to both extend and contract. However, at lower temperatures the performance dramatically decreased, with extension being much more difficult across all sections and multiple sections just being stuck on contraction.

This experiment led to the rejection of the telescopic arm as a feasible form of linear movement, as the material can change but the telescopic arm still has much more mechanical articulations than the stepper motor system, and extreme temperatures can change the precision of the actuation performance.

2.4.4 FRAGILITY AND STRENGTH OF MATERIAL

Another important risk to take into consideration is the possibility of the BreakerBot suddenly breaking. This could happen due to wrong usage, using it too much or maybe a technician not correctly measuring the force that he puts into it. In order to avoid the malfunctioning of the BreakerBot that this would create, different material options are analyzed to employ. Therefore, to choose the perfect material, two different criteria will be taken into account: market analysis and material characteristics.

Firstly, it is important to do a market analysis:

- Price
- Easy to purchase and get delivered
- Simple to manufacture and introduce it into our production chain

Secondly, we need to check the material characteristics:

- Not too fragile but robust enough for a turbine
- Flexible but shouldn't lose shape
- Light (to be easily removable and not obstruct or hinder the movements of the breaker)
- Long life-lasting
- Not Electrically conductive (don't want it to interfere with any other signals or create extra magnetic fields or similar)

All these things could be tested by doing several experiments such as a tensile test and checking the manufacturers characteristics given for each material.

As a conclusion, it was realized the best material would be some plastic reinforced with fiber such as CFRP or Polycarbonate.

2.4.5 TECHNICIAN SAFETY AND AUTOMATIC SHUTOFF

If a technician was to open a panel covering a breaker and try to access the breaker without turning off the microcontroller that controls BreakerBot, there is a possibility that BreakerBot could receive a command to flip that breaker from another technician. This poses a safety risk to the technician as the motors in the breaker actuator (or the current flowing through them) could injure them. Thus, it is very important to ensure BreakerBot is off while technicians service the breaker cabinet. There were two viable solutions to the problem.

The first solution was to add a clear on/off switch. If installing this in a central, highly visible location inside the breaker cabinet, the risk that BreakerBot is on while being serviced is highly reduced.

The other solution is to use a magnetic sensor on the cabinet doors. By attaching a magnet to one door and a reed switch that is integrated into the BreakerBot on the other door, this sensor will detect when the two doors are more than one centimeter away from each other. When the door is open, the BreakerBot will automatically shut off.

While both solutions have their tradeoffs, they would be effective at minimizing the risk of accidental damages. In the one hand, the switch is simple and easier to install, but it doesn't

completely eliminate the risk. In the other hand, the magnetic sensor does completely eliminate the risk, but it assumes that BreakerBot will only operate with the breaker cabinet doors closed, which may not always be intended. *Figure 21* shows a diagram of the previous designs.

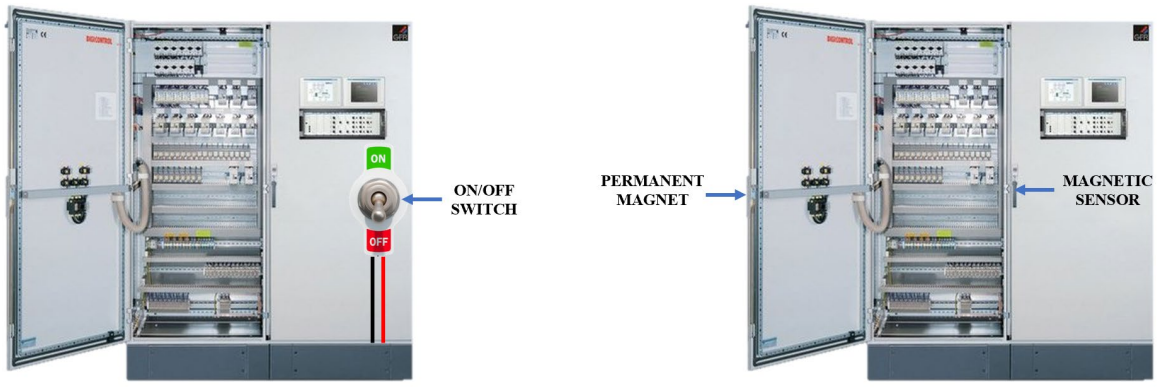


Figure 21. Powerup switch vs powerup magnetic sensor circuit diagrams

The magnetic sensor might be the better solution for the final design as it totally removes any dependency on the technician to manually turn off BreakerBot. This sensor will be very small and low-profile. However, one of the cons of using the magnetic sensor is that it assumes the BreakerBot will only operate when the cabinet is closed, but this tradeoff is worth it. By removing the need for the technician to shut down the BreakerBot, the risk of the design potentially damaging any nearby equipment or technicians is eliminated. Furthermore, RWE confirmed that BreakerBot will always have the cabinet doors closed while in operation.

2.4.6 OVERVOLTAGE

In order to prevent the damage of the electronic components that are going to be used in our bot, overvoltage should be a concern. It is required to clamp the output or shut down the supply when a preset level of voltage is exceeded. This is called overvoltage protection. The main reasons why overvoltage can appear are due to faults from the power supply or from external causes such as those that appear in the distribution lines. An overvoltage condition in a circuit can have several consequences, such as damaging the components or degrading them and cause fire or circuit malfunctions. Therefore, the bot must be prepared for this possible problem, so it does not break and continues with its normal functioning.

When implanting an overvoltage protection circuit, the following characteristics must be achieved:

- Make sure that the excess voltage is not applied to the components of the bot.
- The overvoltage protection circuit should not interfere with the circuit normal operation.
- The overvoltage protection circuit should distinguish between any harmful overvoltage and normal voltage fluctuations.
- The overvoltage protection circuit must be fast enough in his response so that downstream components and power supply do not get damaged.

There are several ways to protect our system from this overvoltage. Generally, most power supplies have implemented an overvoltage protection circuit so that the electronic components are prevented from being damaged. However, this is not always the case, and some possible design to accomplish this overvoltage protection might be useful. For example, the SCR thyristor crowbar, which is an effective, cheap, and simple overvoltage protection circuit. The design is shown in *Figure 22*.

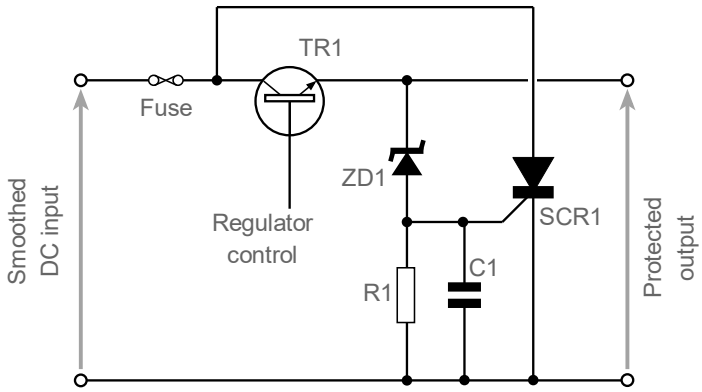


Figure 22. Crowbar protection circuit [21]

During normal operation, the Zener diode does not conduct as it is reverse biased and all the current, which travels through the transistor, appears at the output. When the voltage is high enough that it goes beyond Zener breakdown voltage, this diode breaks down and starts conducting, SCR is triggered, a short circuit is placed and all the current is sunk into the ground, so all the voltage is removed from the protected output.

All in all, overvoltage is a risk to seriously take into consideration, but there are several ways to reduce this risk such as using a power supply with an overvoltage circuit already implemented or implementing an overvoltage protection circuit.

2.4.7 POWER SUPPLY AND POWER LOSS

When talking about reducing risks, the power supply of the device plays an important role, because it is the source of everything happening or not in the bot. Two main ways to feed the apparatus: either using AC or DC. Both AC and DC have their own characteristics and provide different advantages that can be used in different situations. To decide which one to use, there are three common risks to have special care about: technicians having electrical shocks, electronic components being burned and completely loss of power for the BreakerBot.

An electric shock is caused when a current, which is the move of charges from high to low potential, flows though the body when a person meets an electrical energy source. Therefore, the body feels an electric shock mainly because of heating and stimulation of nerves and muscles.

Both currents AC and DC have their tradeoffs when referring to electrical shocks:

- While in DC a continuous muscle contraction is received, in AC current you undergo a series of muscle contraction and electrical pulses that interfere with the nervous system.
- To produce similar effects more DC than AC current is required as the let-go threshold for DC current is higher.
- DC voltage values, which are directly related to current value, are usually lower than AC voltage values.

From prior knowledge, AC power is typically used for high power and long-distance transmission, while DC power is used for lower power items like computers and other devices. AC flips polarity very often. Most electronic devices have been designed with particular polarity conditions in mind. For example, chips require a constant flux and flow

of electrons to work and store data. Making an electronic circuit with AC current make things more challenging.

To avoid is having our bot stop being powered, the first option was using batteries that would have enough supply for our device while wind is not providing energy. Then, these batteries would be recharged once our wind turbine starts the generation again. This idea was at first the primary option but then, after talking with the company and learning from prior devices, the possibility to connect the apparatus to the electric network appeared. Even though wind turbines are located up in mountains, they have a direct connection with the power grid to supply the power generated.

All in all, the final design will be powered with a DC power source as it works better with electronic components, gives lower current values and can be connected to the power grid.

2.4.8 PROPER BREAKER DETECTION

For physical interaction with the breaker, different detection mechanisms were tested in a simulated environment that resembled what the inside of the cabinet would look like. This involved testing with different temperature conditions and lighting conditions. The three detection methods that were quickly tested were using a camera, a PIR sensor, and a Hall-effect sensor.

2.4.8.1 PIR Sensors

Research:

- As temperature increases (around body temperature 37°C) the range of PIR sensors decreases.
- Lighting that quickly change in temperature can cause a false trigger.

Testing:

To test the feasibility of the PIR sensor a PIR sensor was connected to an Arduino and checked if its performance decreased significantly with the sensor being in the freezer and in the oven temperatures. The freezer was at about -18°C and the oven was set to 45°C for 30 minutes. Then, the PIR sensor was placed in front of a breaker and manually flipped it.

This experiment ran twice, once in room lighting and then again in with the lights turned off.

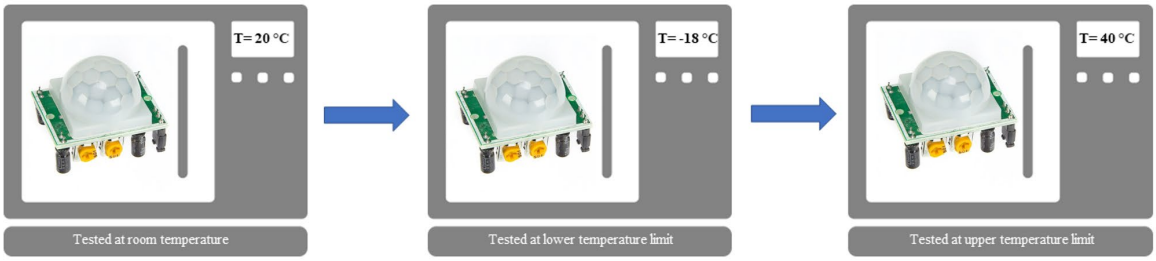


Figure 23. Testing PIR sensor at boundary temperature limits

Results:

Neither lighting nor temperature had a huge effect on the readings. The Arduino was still able to tell when a high input was received and when a low input was received.

2.4.8.2 Camera recognition

Research:

- Requires lighting or to have night vision.
- Requires image recognition algorithm.

Testing:

To test this, a regular camera was used and checked if the camera was able to see the breaker flipped in room lighting and then with the lights turned off.

Results:

When the lights were off the camera was not able to show when a breaker flipped. However, this was resolved by using the integrated flash. Meaning a lighting source would have to be add to the design if a camera were to be used.

2.4.8.3 Hall-effect sensor

Research:

- Operating temperature -20 to 40 °C (within the range expected to work in).
- Expected magnetic field variations.

Testing:

To test the feasibility of the Hall-effect sensor, a sensor was connected to an Arduino and checked if its performance decreased significantly with the permanent magnet being in the freezer and in the oven temperatures. The freezer was at about -18°C and the oven was set to 45°C for 30 minutes. Then placed the permanent magnet close to the Hall sensor and observed differences.

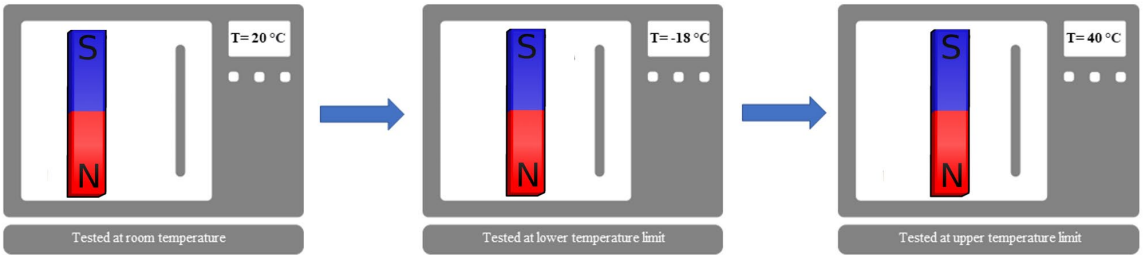


Figure 24. Testing Hall-effect sensor for magnet at boundary temperature limits

Results:

One of the main characteristics of permanent-magnet materials is they have temperature dependent properties. In fact, the residual magnetization and coercivity of magnetic materials decrease as temperature increases. *Figure 24* shows B-H magnetization curves for different temperatures in neodymium-boron-iron.

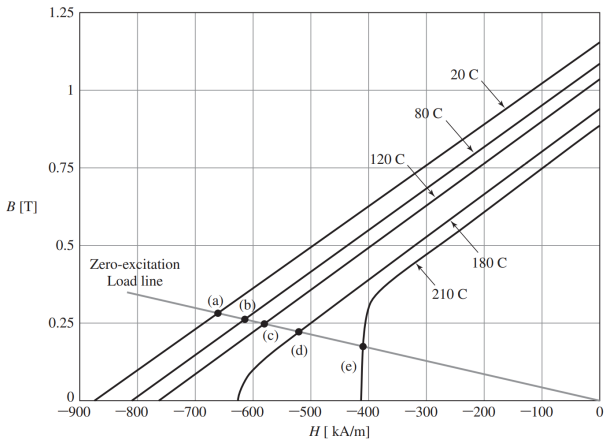


Figure 25. Magnetization curves for neodymium-boron-iron material with temperature dependence [22]

The temperature (at the range tested) had small effects on the analog input Arduino was reading. Cold improves the magnetic property of the permanent magnet and its strength increases. Conversely, when the same magnet is exposed to high temperatures, the

magnetics domains were misaligned, causing magnetism to decrease. However, the Arduino was still able to detect and differentiate between a high and a low input.

2.4.8.4 Conclusion

After testing and considering both the timeline and resources for this project it was decided that the PIR and Hall-effect sensor are more adequate and impeller to implement as, even though magnetic sensor reading values changes with temperature, when connecting to a digital input, both work well within the environmental given constraints.

Chapter 3. HARDWARE

3.1 DESIGN-INTEGRATED DEVICES

3.1.1 ARDUINO UNO

3.1.1.1 Function

Arduino is an open-source electronics creation platform that allows to create different types of microcomputers from a single board for many types of use. The Arduino board has a Atmel AVR (ATmega328p), a single-chip, high-efficient and performance microcontroller. The microcontroller is a programmable integrated circuit that stores and runs instructions. Therefore, Arduino acts as an interface between hardware and software, allowing communication between the user and the breaker position. *Figure 26* shows Arduino Uno R3 board, which is the model used for the final design.

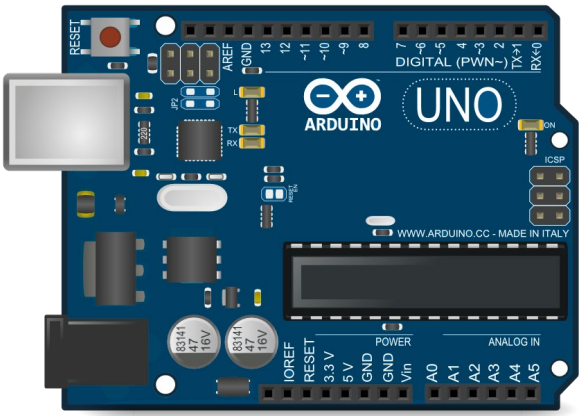


Figure 26. Arduino Uno R3 [10]

3.1.1.2 Rationale

Arduino has been used because it is a device capable of solving all communication problems between the detection and actuation of the breakers, as well as being simple, flexible and intuitive to program.

3.1.1.3 Connection

Arduino has multiple communication interfaces, including: UART, SPI, I2C and USB connection. It also has general purpose pins, as well as power supply at different levels and ground connections, which have been used to interface with the rest of the hardware in the design.

3.1.2 TOUCH SENSOR

3.1.2.1 Function

The function of the touch sensor system is to ensure that the control cabinet door has been properly closed and that the designed system can resume normal operation. If the Bot is correctly positioned, the touch sensor will conduct current and let an LED light shine.

The model of touch sensor to be used cannot be known with certainty as testing depends on the closure of the cabin door.

3.1.2.2 Rationale

The main reason why the touch sensor has been chosen is because, although there are several other sensor options, the only one that can give me a sure guarantee that the closing has taken place properly is the touch sensor, as it will allow BreakerBot to operate as long as the control cabinet door is perceived to be closed.

It was therefore decided to connect an LED in addition to the contact sensor so that the technician can check whether the door has closed correctly. It is important to enhance that it was also taken into account that there was a problem with the first idea of the LEDs. It is not useful to have the LED light on when BreakerBot is correctly positioned. This will only cause the lights to constantly go out and unnecessary use of electricity. Therefore, one solution would be negating the LED input so that only the LED light is on when there is a misplacement.

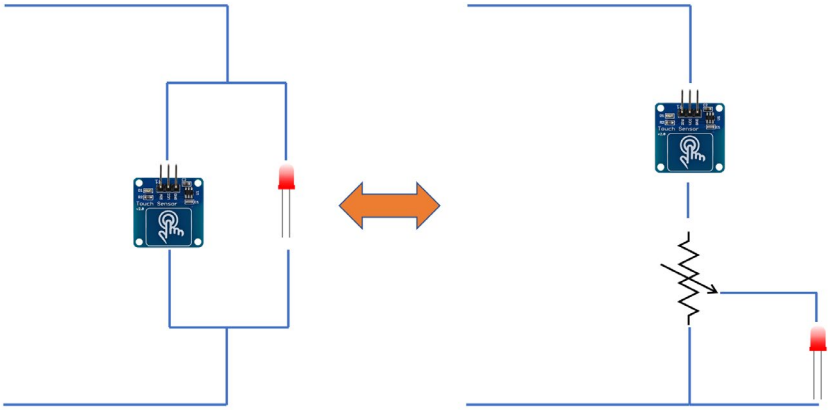


Figure 27. LED schematic

The two designs shown in *Figure 27* are a valid option for implementation in the control cabinet door. However, an alternative design was chosen, similar to the previous ones, which appears in more detail in *Section 3.1.2.3*. It consists of controlling the behavior of the LED through lines of Arduino code in such a way that when the sensor is touched the LED is Off and, consequently, turns on when it loses contact with the sensor.

3.1.2.3 Connection

The connection formed by the touch sensor and the LED, shown in *Figure 28*, is intuitive: it consists of powering the touch sensor on one side and connecting its last leg to the Arduino to send the perception of pressure when it occurs. On the other hand, the LED circuit is powered by a pin that will be the one that we will configure as HIGH in the code when the Arduino does not receive pressure from the sensor, i.e., the cabinet is open.

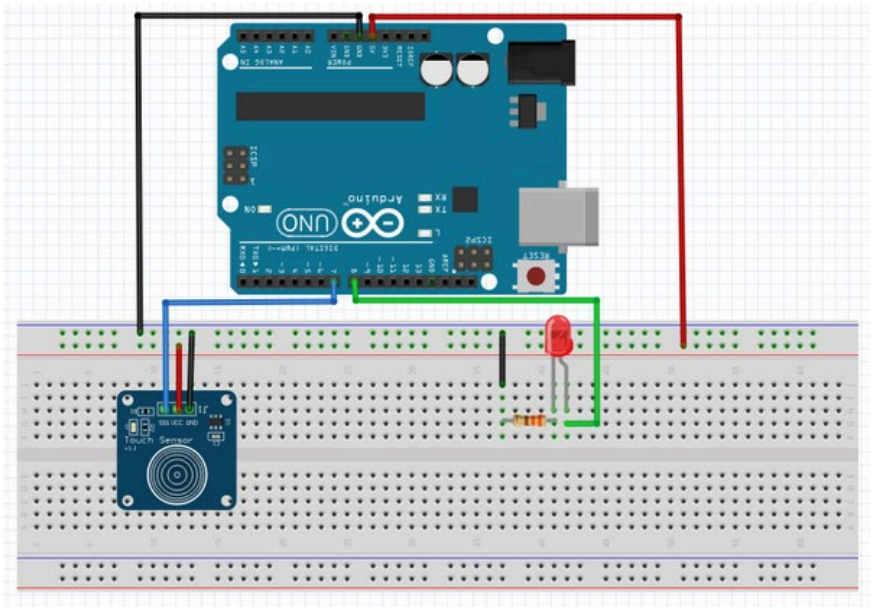


Figure 28.Wiring of the touch sensor and LED circuit [19]

3.2 BREAKER DETECTION

3.2.1 HALL SENSOR

3.2.1.1 Function

This electronic device detects the Hall effect and provides a measurement of the magnetic field from a permanent magnet. The sensor output voltage is directly proportional to the magnetic field strength. The model chosen for the final design is the AH49E Hall-effect sensor, shown in *Figure 29*. External filters to this sensor are not necessary as the integrated circuit gives a noiseless output.

Figure 29. AH49E Hall-effect sensor [18]

3.2.1.2 Rationale

Since the rest of the system is only useful when a breaker is tripped, it is crucial to accurately detect flipped breakers to avoid both false positives and false negatives.

During the design process four detection methods were explored: physically connecting to the breaker to monitor current, a camera with video recognition software, an IR beam with reflectors along the length of the panel, and the Hall-effect sensor, which appears in the final design.

Although monitoring the current through the breaker would give the most accurate data, this method was disproved fairly quickly to avoid any potential issues/additional faults that may occur by altering the circuitry in the tower and to minimize the additional burden this system would put on a technician working in the tower. By connecting to the breaker, additional responsibility on a technician servicing or replacing a breaker appeared to remove the connection and properly reinstall it.

Video recognition software was also ruled out, as although it would result in the smallest footprint inside the cabinet, the increased difficulty to expand the system, the complexity of the accompanying software, and the requirement to add lighting into the cabinet made video recognition infeasible for this project. Software like this would require declaring the specific areas of the video feed watched for a tripped breaker and attempting to add more breakers would require substantial software updates and likely the addition of a second camera.

An IR beam with reflectors on each breaker level was the most promising of the alternatives but had similar pitfalls to the others, expansion would be difficult and additional work would be placed on a technician replacing a breaker. Using a single IR beam pointed through the area where tripped breakers would be, an affixed reflector on each breaker would be required and use the reflected IR beam to measure the distance to the breaker. With that distance, it could have been calculated which breaker had tripped using predetermined measurements. However, to expand this system would require additional measurements to determine the distance to the new breaker and would have required an additional IR beam should the breaker be on a different panel. This method would also limit functionality in the event

multiple breakers trip at the same time, as only the breaker closest to the IR beam would be detectable.

Finally, the effect-Hall sensor was the most flexibility for expansion, the least burden on the technician, and it is reasonably simple for a project of this length, while keeping a really accurate detection precision. Moreover, this sensor offers a fast response that is not affected by dust, and it has a very long service life.

3.2.1.3 Connection

The wiring of the sensor is simple: the supply pin is connected to the appropriate terminal block corresponding to the Arduino the sensor interfaces with. The ground pin is connected to another series of terminal blocks which are connected to any of the Arduinos' ground sources. The last pin is connected to either an analog or digital input of the Arduino board. The previously explained connection is shown in Figure 5.

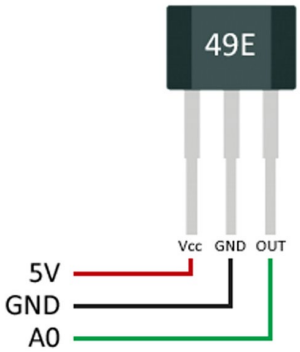


Figure 30. Wiring of the Hall-effect sensor [17]

3.2.2 MAGNET

3.2.2.1 Function

The role of the permanent magnet is to give a sufficiently strong magnetic field in an air gap so that the Hall effect sensor can detect the magnetic field as it approaches. The model chosen for the final design is rare earth Neodymium magnet, shown in *Figure 31*.

Figure 31. Rare earth Neodymium magnet [24]

3.2.2.2 Rationale

To choose the type of magnet for the sensor, it is necessary to check their properties to ensure, for example, that the magnetic field is strong enough for the sensor to detect it.

Thus, there were two possible types of magnets: rare earth magnets or regular magnets. Comparing both types of magnets, it was realized rare earth magnets, which are composed of alloys of rare-earth elements, have higher energy product, stronger magnetic output, and higher coercive force than regular magnets as ceramic or alnico magnets.

Within rare earth magnets class, there are two main types: Neodymium or Samarium Cobalt magnets. While both magnets have high magnetic performance and high resistance to demagnetization, Samarium Cobalt magnets are more stable at high temperatures and have higher resistance to corrosion. However, the magnet chosen for the final design is a Neodymium magnet, which is less fragile (fits better with the constant vibration environment of a wind turbine) and has a lower cost than Samarium Cobalt magnets, while maintaining a good performance in the temperature operating range.

3.2.2.3 Connection

To detect the breaker tripping, the magnet is placed on the left side of the breaker switch (*Figure 32*). Whenever the breaker is operating normally, the sensor cannot detect the magnet. When the breaker trips, the magnet will flip into the detection area of the sensor. Every breaker under the coverage of BreakerBot will have one magnet and magnetic sensor. This will allow to keep the area where actuation occurs (the bottom or “off” position of the breaker) clear for the actuation system to operate.

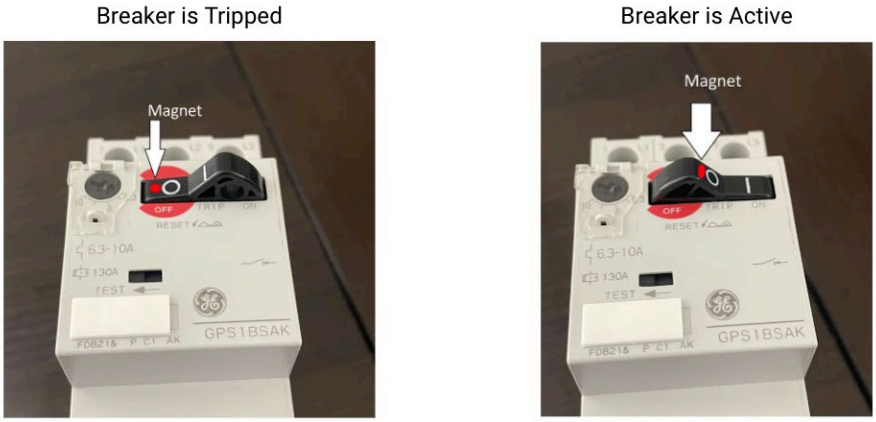


Figure 32. Position of the permanent magnet in the breaker

3.3 BREAKER ACTUATION

3.3.1 RELAYS

3.3.1.1 Function

The purpose of the two relays is to act as a driver and control when the motor should either rotate or stop. The model used is a two-channel relay module DC 12V 10A low/high trigger, which can be seen in *Figure 33*. To make the circuit work, two relays, the DC motor, the Arduino board, and an external power supply for the motor are required. *Figure 34* shows a diagram of this circuit.

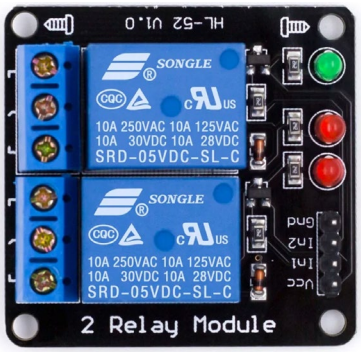


Figure 33. Two-channel relay module DC 5V 10A low/high trigger [13]

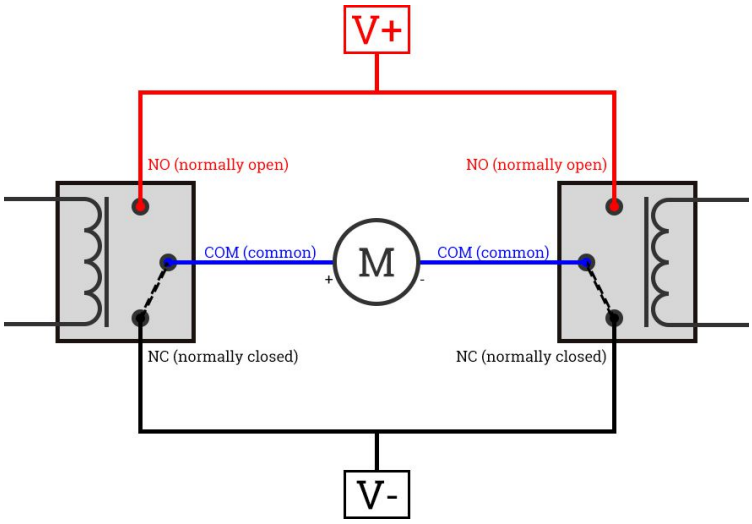


Figure 34. Relay module circuit diagram [14]

3.3.1.2 Rationale

An alternative to this circuit was to use an H-Bridge with four switches, as shown in Figure 4, to control the different states of the motor. However, the H-Bridge requires a much more precise control of its switches, as it is necessary to open a switch and after a small delay close the other one. If both switches in a phase leg are on at the same time, shoot through fault can occur and cause damages to the power supply or the circuit itself.

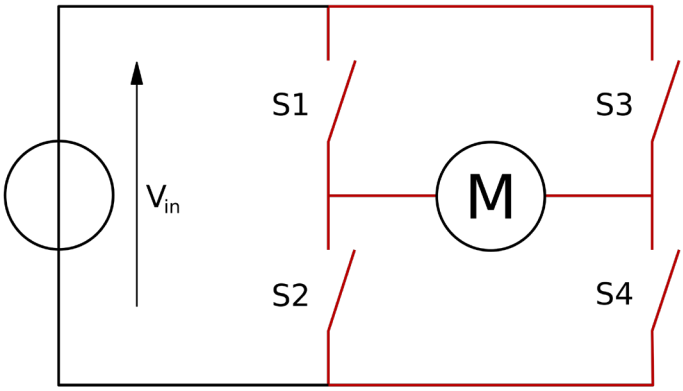


Figure 35. DC motor controlled by an H-Bridge [15]

Therefore, using two-relays circuit properly, a single relay acts as a full phase leg of the H-Bridge and, even though a relay can be stuck in one state, it cannot conduct current on both terminals, avoiding a shoot-through fault. Furthermore, the module circuit is optocoupled, providing galvanic isolation and low-voltage devices are twice protected from high-voltage circuits.

3.3.1.3 Connection

The connection of the relay module circuit with Arduino is simple. The positive and negative terminals of power supply for the motor are connected to the same side of the two relays (either normally open or normally closed) and the common terminals are connected one to the DC motor, in accordance with the previous connection. Then, the other side of the relays is connected to the microcontroller, providing 5V to power it and pulling to ground voltage the two input pins. *Figure 36* represent the wiring connection and the possible motor's outputs depending on the states of the relays.

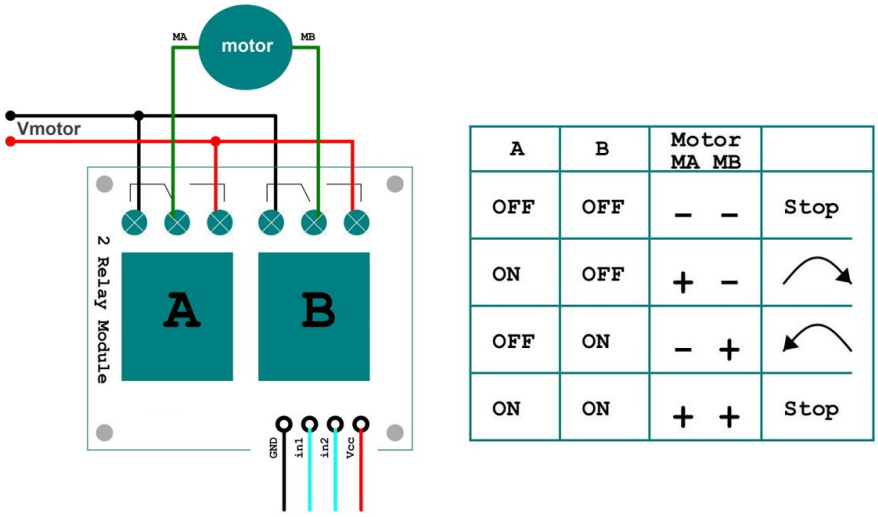


Figure 36. Wiring of the relay module circuit [15]

3.3.2 LINEAR ACTUATOR

3.3.2.1 Function

A linear actuator is a device capable of moving objects in a linear direction, after transforming rotary motion into linear motion. The rotary motion is first performed by an electric motor that can rotate thousands of revolutions per minute. This high angular velocity is reduced by a gearbox, sometimes with a "100:1" ratio to increase the torque to be used to rotate the spindle. The, the leadscrew turns, resulting in a linear movement of the nut. *Figure 37* shows the operation of a linear actuator.

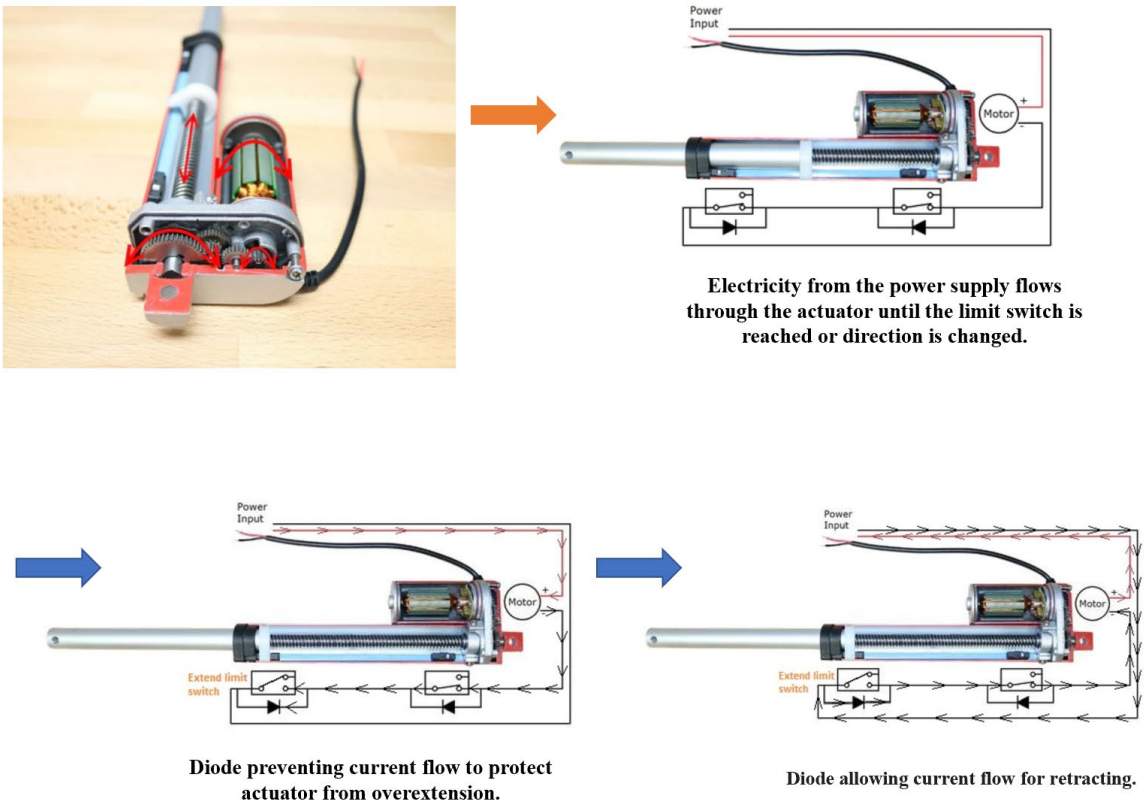


Figure 37. Linear actuator operation

The function of the linear actuator in this project is one of the most important since it is responsible for resetting the switches with its actuation. The model used for the final design of the BreakerBot is a PA-14 mini linear actuator model shown in *Figure 38.*

Figure 38. PA-14 mini linear actuator [36]

3.3.2.2 Rationale

When sourcing actuators to use for the project, the actuator researched had to be compact in size and able to produce at least 18 lb*inches of torque, per the datasheet of the breaker, to correctly flip it back. However, all linear actuators are described with their force produced in lbs. To guarantee getting the right actuator, force and torque had to be compared by using *E.1.*

$$E.\ 1 \qquad \vec{\tau} = \vec{r} \times \vec{F} = \begin{vmatrix} \bar{\imath} & \bar{\jmath} & \bar{k} \\ x & y & z \\ F_x & F_y & F_z \end{vmatrix}$$

where the vectors i,j,k , the distances x,y,z and the forces F_x, F_y, F_z represent the operations in the x,y,z axis directions, respectively. In the case of the system under study, it would look like this:

$$\vec{\tau} = \begin{vmatrix} \bar{\imath} & \bar{\jmath} & \bar{k} \\ x & 0 & 0 \\ -F * cos\theta & 0 & -F * sin\theta \end{vmatrix} = x * F * sen\theta\ \bar{\jmath}$$

Since the axis whose torque is to be studied is the y-axis, the resulting force to reset the switch depends on the orientation given to the linear actuator. Thus, *Figure 40* shows the two positions considered.

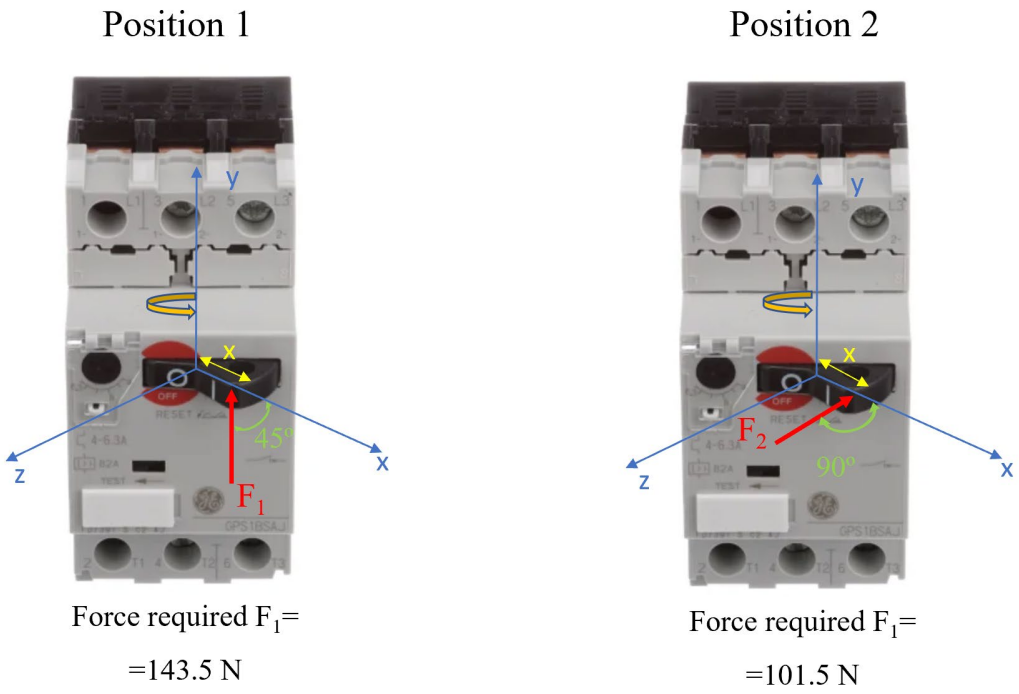


Figure 39. Linear actuator force

On the one hand, as the angle in position 1 is 45º, knowing that the torque required is 18lbs*in (2.03 Nm) and the distance along the x-axis is 2 cm (measured the radius of the switch), the force to reset the breaker would be F_1= 2.03/ (0.02*sin 45º) = 143.5 N = 32.3 lbs.

On the other hand, in position 2, we have the same torque and distance as conditions, but this time the angle is 90º. Thus, the force to reset the breaker would be F_2= 2.03/ (0.02*sin 90º) = 101.5 N = 22.8 lbs.

When the linear actuator was ordered, it was chosen in such a way that it would be able to make the force required in the worst case of the two cases studied, that of position 1 (32lbs). Therefore, the linear actuator that was purchased was rated at 35 lbs of force. Moreover, the obtained load of 35lbs allows to get the maximum speed out of our linear actuator according to the graphs given by the data sheet. This can be seen in Figure

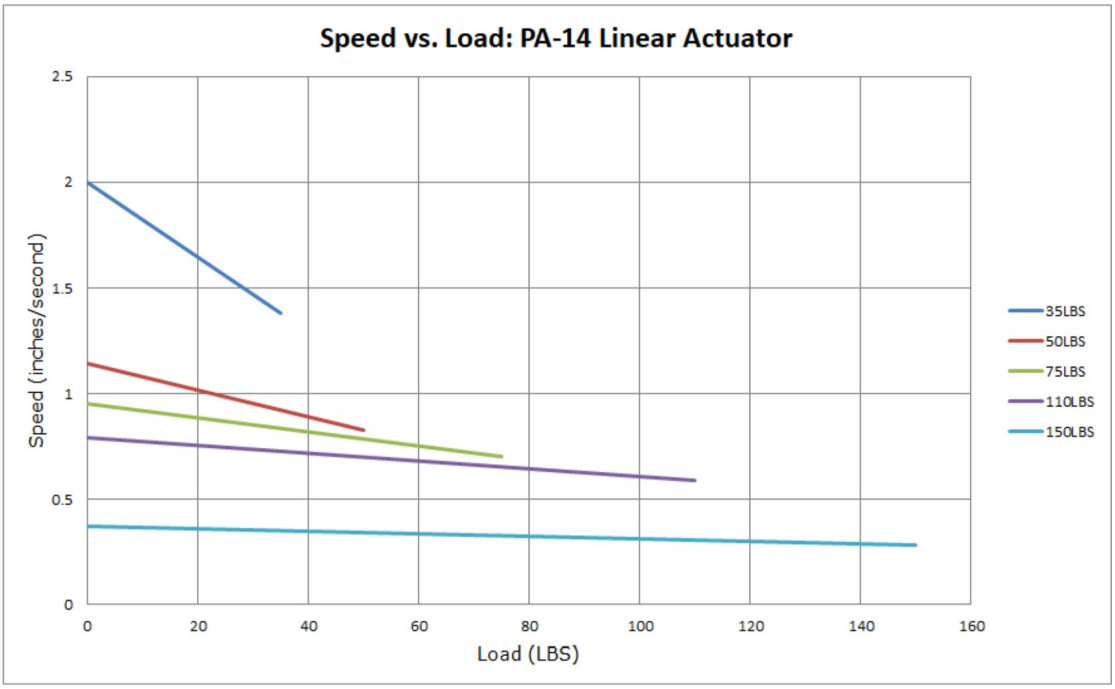


Figure 40. Speed vs Load for linear actuators with different forces [36]

In this way, it could be ensured that the switch would be activated when linear actuation occurred, regardless of the orientation of the linear actuator.

3.3.2.3 Connection

This linear actuator works by connecting it directly to the power supply, and its direction will be output or input according to the polarity of its input terminals. As in this project we want to control the actuator behavior and actuation times, the connection to the microcontroller is done through relays, as shown in the *Section 3.3.1.3.*

3.3.3 STEPPER MOTOR

3.3.3.1 Function

Stepper motors are DC motors whose rotation is done by steps. They are controlled by applying DC pulses to their internal coils to reach a specific position in reference to the starting point. As steppers motors have a magnetic core surrounded by two coils, by controlling the current flowing through the coils, the motor can move certain steps. Figure 41 shows how the magnetized core changes its position depending on which coil is energized.

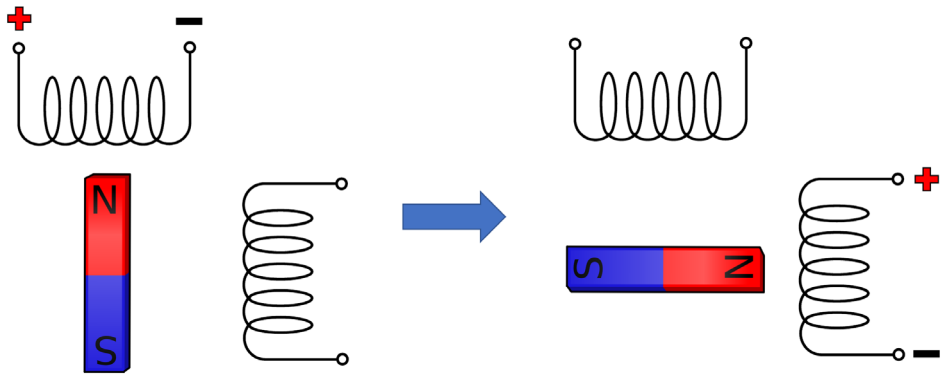


Figure 41. Magnetized shaft being attracted to energized coil

Furthermore, if a smoother rotation is required, micro step controller can be used by changing the ratio of current flowing through both coils to draw the motor to a position between the coils.

In this project, the function of the stepper motors is to move the linear actuator horizontally to the exact position of each breaker. The model chosen for testing is a 12 V DC D8-MOTOR80 stepper motor (*Figure 42*), which would not be the motor used in the prototype to be delivered to RWE but serves to simulate what operation would be like with a more expensive model.

Figure 42. 12 V DC D8-MOTOR80 stepper motor [33]

3.3.3.2 Rationale

To reach the proper position of the desired breaker, two types of motors can be used for accuracy and precision: stepper motors and servo motors. Although both are a good choice for the final design, they have some differences that distinguish them from each other.

The main advantage of servo motors is that they provide high levels of torque at high speed, while stepper motors can only provide high torque level at low speed. Furthermore, the speed of a servo motor is higher than the stepper motor and servo motors have lower vibration and resonance levels. However, as stepper motors have a high pole count, they can give a precision drive control, while servo motors require closed loop control with feedback from encoders and often a gearbox. Thus, servo motors systems are more expensive than stepper motors and they are also more mechanically complex, which means more maintenance costs.

For the requirements given in this project, a stepper motor can be used as it is not necessary an extremely fast response for the designed system. If the stepper motor is set under 1,000 revolutions per minute, their torque response will be excellent, as shown in *Figure 43*.

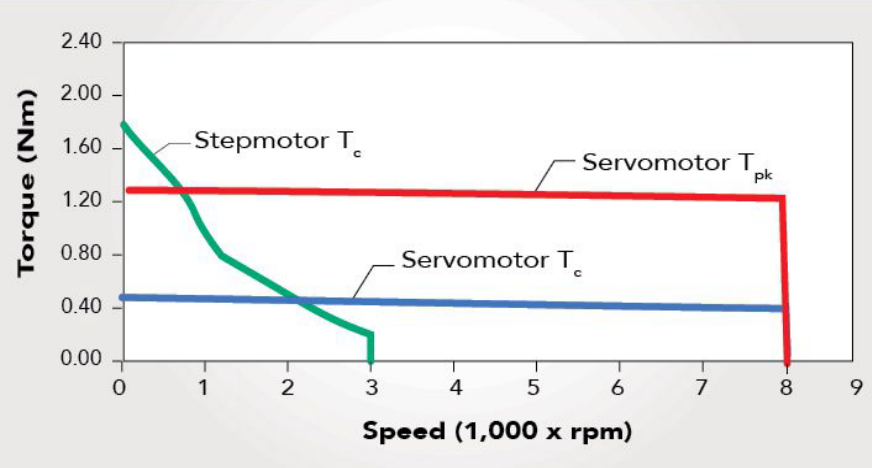


Figure 43. Torque output (Nm) vs speed (rpm) for stepper and servo motors [35].

3.3.3.3 Connection

The connection of the stepper motor with the Arduino is done by using a A4988 microcontroller, whose function is explained in *Section 3.3.4*. The motor cables are connected, through the breadboard, to the A1, A2, B1 and B2 pins that are the two coils (A

and B) that the stepper motor has. The power supply for the motor is also connected through the microcontroller which would decide the coil to be energized. The rest of the connections shown in *Figure 44*, are essential for the microcontroller to be powered and control the direction and steps of the motor properly.

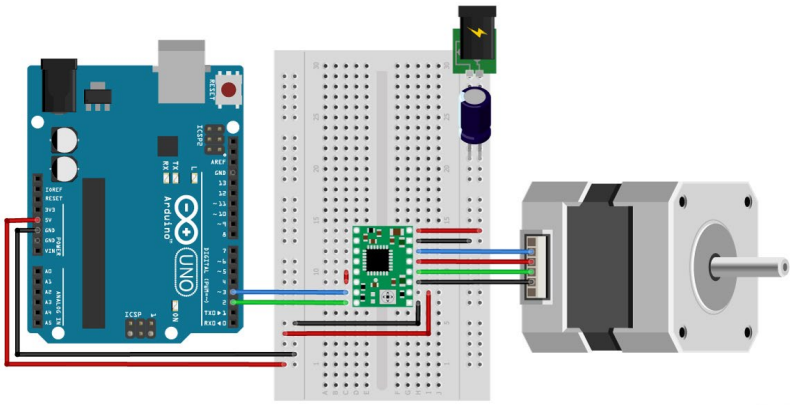


Figure 44. Wiring of the stepper motor with the driver [34]

3.3.4 DRIVER A4988

3.3.4.1 Function

To control the stepper motor, a dedicated bipolar stepper motor controller (A4988) will be used. When this inexpensive controller is connected to the stepper motor, a heatsink is commonly sticked to the controller and a decoupling capacitor is also required to prevent peak currents. *Figure 45* shows the pinout of the A4988 module, which are explained below.

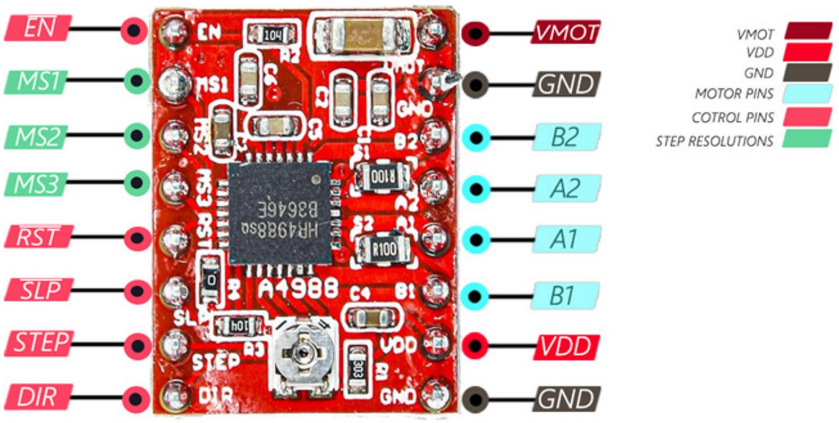


Figure 45. Pinout of the A4988 module [34]

- **VMOT:** the motor 12 V DC supply voltage given by the power source.
- **GND:** the motor supply voltage to ground.
- **B2, A2:** the connections to the second coil of the stepper motor.
- **A1, B1:** the connections to the first coil of the stepper motor.
- **VDD:** the 5 V DC logic supply given by Arduino.
- **GND:** the ground logic supply given by Arduino.
- **ENABLE:** the pin turns on the motor.
- **MS1, MS2, MS3:** micro-step selection pins.
- **RESET:** the pin sets the motor to default settings.
- **SLEEP:** the power states control pin.
- **STEP:** the steps control pin.
- **DIR:** the motor direction control pin.

3.3.4.2 Rationale

There are many reasons to choose A4988 controller for the stepper motor. Firstly, all the stepper motor logic is contained in the controller, freeing the Arduino to do other things. Then, this microcontroller has reduced connections which makes things easier to control multiple stepper motors. Finally, micro stepping is easy as it is controlled by different configurations of MS1, MS2 and MS3.

3.3.4.3 Connection

The connection was previously shown in *Figure 45*, where it can be seen how the A4988 only needs two inputs from Arduino to control the stepper motor. Furthermore, a 100 µF capacitor was used to decouple the power supply.

One more thing to consider before running the circuit is to set the current that flows through the motor's coils by using the potentiometer of the microcontroller to make sure not to burn off the motor. To do this, firstly, sleep and reset pins must be bridged and VDD and GND need to be connected to the Arduino board. Then, it is necessary to calculate the reference voltage by using equation *E.2*.

$$E.\ 2 \qquad V_{ref} = 8 * I_{motor} * R_{sens}$$

Where the reference voltage is the maximum motor current by the current sensing resistance, multiplied by 8. As the maximum motor current is 800 mA and the sensing resistance is 0.068 Ω, the reference voltage is 0.44 V. By measuring between ground and the potentiometer with a multimeter, the potentiometer's voltage can be set to measure the calculated reference voltage.

3.3.5 HC-06 ARDUINO CONNECTION

3.3.5.1 Function

Remote communication with Arduino will be used for on-site testing. As BreakerBot is to be placed inside the control booth, the computer that will be used to control the operation of the system and simulate the response by the technician will need to be connected to the device wirelessly.

3.3.5.2 Rationale

The use of Bluetooth communication with Arduino is to be able to remotely control both the detection and actuation of the system without having to implement a wired connection in the wind turbine control cabin, which is in the nacelle.

Thus, at the bottom of the tower there would be a computer that could be accessed remotely and through which the entire process would be carried out.

In addition, and as explained in *Section 6.4*, the BreakerBot control application should be installed in the control center and communicate remotely via SCADA system. However, there is no access to the communication center for the installation of the application or to receive data via SCADA.

3.3.5.3 Connection

The Bluetooth module HC-06 has 4 pins to establish the connection. The power supply (VCC) pin, which is connected to the 5V pin of the Arduino, the ground (GND), which is connected to the GND pin of the Arduino, the RX receive pin, which is usually connected to the Arduino transmit pin (TX) and finally the transmit pin (TX), connected to the Arduino receive pin (RX), usually. This connection explained above is shown in *Figure 46.*

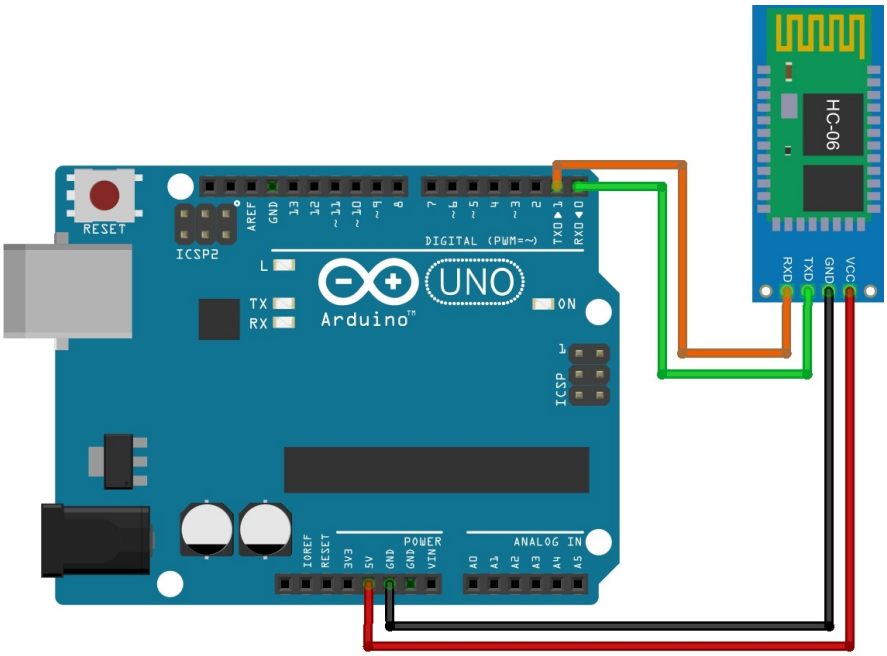


Figure 46. Wiring of the HC-06 to Arduino [37]

Chapter 4. SOFTWARE

The designed communication system consists of the elements shown in *Figure 47.*
On the one hand, to communicate with the technician web application, Arduino controller will clean up the data and send it to the technician portal backend, which will be done with MATLAB. On the other hand, Arduino communicates with the Raspberry Pi which will be configured to act as an OPC-UA server for transmitting data endpoints between systems. RWE will configure their systems to receive this OPC feed. To get the data from the Arduinos, each Arduino is connected to the Raspberry Pi via USB.

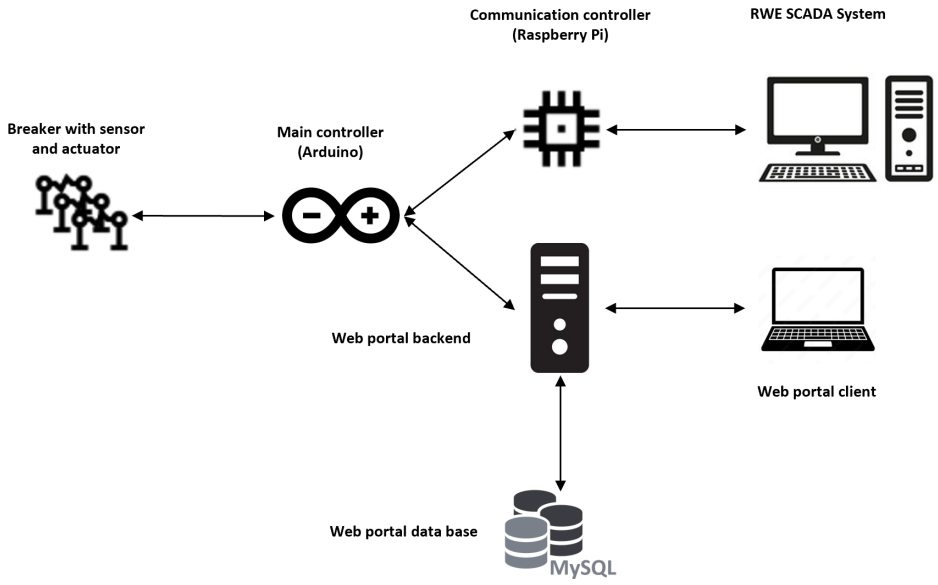


Figure 47. Overview of the Communication System

The web portal for technicians consists of a MATLAB backend using App designer, a React frontend and a MySQL database. The backend serves to receive data from the Arduino, update the data in the database, and then serve this data to the frontend through various API endpoints. It will additionally accept requests from the frontend to reset a breaker and then send those on to the Arduino to be acted upon. An initial concern appeared while designing this system was compliance with the NERC cybersecurity standards, as a fully custom solution would then require access management, encryption,

and certification from NERC. However, because connectivity is achieved through RWE provided networking and the web portal runs off an already certified RWE server, the system will be compliant so long as all data sent and received is done through devices on RWE's network.

4.1 ARDUINO IDE

The task of Arduino's code (*Appendix II*) is to run a code that is divided into two parts:

1) The setup() function, which will only be executed once, after each powerup. It is used to initialize pin modes, variables, etc.
2) The loop() function, which will loops consecutively, enabling the program to change the guidelines to control the Arduino board.

The void set up will define all the pins that are being used and what they are connected to. Thus, we will distinguish between inputs and outputs, being the sensor the only input, and the responses on the relays, the LED, the step and direction of the stepper motor, the outputs. Also, in this part of the program is where the LED is initialized to be "on".

The void loop will be where the communication with MATLAB will take place. Arduino will send the digital value of the LED. The LED value will be "on" as long as the breaker is not turned off. The short-range position sensor will sit next to the lever to detect its presence in the "on" or "functional" position, using negative logic to detect a tripped breaker (i.e., a low or "0" signal indicates that the breaker has tripped). The signals from the position sensors will be sent into the detection controller where the aforementioned negative logic will occur.

MATLAB will then send back a signal on how to act on the system. Arduino first asks if it can read from a serial port, and then reads. Moreover, Arduino checks if it has received a signal to act from the user via MATLAB or not.

Only if a signal to actuate has been received, the stepper motor platform will move away from the stepper motor a certain distance depending on the number of turns, which is given by the number of pulses multiplied by 1.8º/pulse. For example, in the attached code (*Appendix II*), 1000 pulses have been set, which would result in 1800º which is the same as

5 turns of the stepper motor. The number of steps to be taken will be adjusted according to the position of each breaker.

The other parameter is the speed of the motor, which will be determined by the frequency of the pulses sent to the "step" pin. A pulse is the result of having a signal at high a little and then setting it to low, repeating this process. By changing the delay between two pulses, the frequency of those pulses, and therefore the speed of the motor, is changed.

Once the stepper motor reaches the desired position, the Arduino commands the linear actuator to reset the switch, combining the relay positions. In this way, the linear actuator exits, actuates and returns to its initial position. Finally, the stepper motor returns the platform to the initial position.

Since each sensor will produce a binary signal, it is intended to assign each breaker a specific bit in a simple unsigned integer, so that a bit mask can be put at the controller input to detect which breaker has jumped. Then, a similar binary encoded decimal can be passed to the primary controller for processing. This allows the greatest amount of flexibility to add additional switches to the system (up to 64), as existing code fragments can simply be reused with minor modifications to accommodate more inputs.

4.2 PORTAL BACKEND

4.2.1 MATLAB

In this project, the objective of using MATLAB is to create a graphical user interface (GUI) similar to that of a control panel so that technicians can have control over the resetting of the switches from the application. For this purpose, within MATLAB, App Designer was used, which is an interactive development environment for programming the behavior of an application. The developed code appears in the *Appendix III.*

The code, in addition to the definitions of the properties of the designed interface, presents 5 main functions:

- **Startup function:** this function is compiled each time the application is opened. It establishes the connection to the Arduino through the port. In addition, the desired characteristics of the elements are initialized for the start of the program.
- **Connection function:** the function initiates the communication, previously defined in the Startup function, with the Arduino. In addition, it reads through the port the digital values of the LEDs, which represent the states of the switches, sent by Arduino and classifies them. Thus, as long as the states of the switches are "On", it will simply make graphs showing this position of the switches. As soon as a switch becomes "Off", it will show on the screen that the state of this switch has changed and this change will only be visible in the graph of the off switch.
- **ResetBreaker function:** this function will come into play when the user tries to change the position of a switch. As technicians only have access to reset switches, and not to turn them off, if they try to change their position to off, the program will not let them. If, on the other hand, the breaker has been turned off, when the user presses the switch to turn it "On", the breaker will be reset.
- **Disconnection function:** when the user presses the disconnect switch, the program will close MATLAB communication with Arduino.
- **Close function:** before closing the program, this function will not only close the communication between MATLAB and Arduino but will also delete the communication established in the function.

4.2.2 PORTAL DATA BASE

In this project, MySQL database application is used for database management. Database management systems (DBMS) are software systems used to store and process queries on data. A DBMS serves as an interconnection between a user and a database, allowing users to access and modify data in the database. Therefore, the function of MySQL is data storage. The choice of this application for the storage of data on the operations performed on the system by the operating technicians is based on three main reasons: high data protection, top performance, and ease of communication with the software.

On the one hand, MySQL can meet the performance needs of different systems while accommodating the needs of some business-critical systems. Furthermore, as for database authentication, MySQL provides powerful mechanisms to ensure that only authorized users have access to the database server, and it is possible to block users down to the client machine level.

On the other hand, the communication between MySQL and MATLAB and vice versa is straightforward and is limited to include a couple of additional lines to the final code. As this project simulates the situation in the control cabin of a wind turbine, there was really no information to collect in the database and data storage via MySQL has not been included in the code. However, in the following example, we simulate having a MySQL design, which contains data about the circuit breakers, their number of incidents and the total number of monthly incidents. Assuming to be connected to a MySQL database that has tables named "incidentVolume" and "monthlyIncidents". The "incidentVolume" table contains the column names for each breaker. The "monthlyIncidents" table has the column names "Breakers" and "IncidentsTotal". Here, it is explained how to import data into MATLAB from MySQL, modify it and return it to this platform, for any operation that the technicians or RWE may wish to perform.

- **Create Database Connection:** Starting by creating a connection to a MySQL database using the data source name, username, and password.

```
datasource = "MySQLData";
username = "rwe";
password = "BreakerBot";
connection = mysql(datasource,username,password);
```

- **Import Data from Operations:** import incident data using the database connection. After reading the data from MySQL it can be modified and updated from MATLAB, for example, the total number of Breaker 14 incidents is being calculated.

```
tablename = "incidentVolume";
data = sqlread(connection,table);
total = sum(data.Breaker14)
```

- **Update in MySQL:** Retrieve the name of the breaker from the data of the collected incidents, define, if this has not been done previously, the names of the data columns and create a table that stores the data to export in MATLAB. Furthermore, revise the status of the AutoCommit database flag to check whether the data import can be performed or not. Finally, using the "toystore_doc" catalog insert the new incidents of the breaker specified in the monthly incidents table.

```
breaker = data.Properties.VariableNames(4);
colnames = ["Breaker" "IncidentsTotal"];
results = table(breaker, total, 'VariableNames', colnames);
connection.AutoCommit;
tablename = "monthlyIncidents";
sqlwrite(connection, table, results, 'Catalog', 'toystore_doc');
close(connection);
```

4.3 PORTAL FRONTEND

Once the MATLAB and Arduino functions are implemented, the design and coding part of the software would be finished and ready to control the behavior of the BreakerBot hardware. The following is a description of how the application works with the system.

Initially, when the script is compiled, the graphical interface that appears looks like *Figure 48.* The interface is divided into two parts: the left panel and the right panel.

The right panel is purely for receiving information. For example, in the design you can see graphs of power generation by RWE wind turbines at different scales: worldwide, USA, Texas or Pyron farm.

The left panel is the switch cabinet control panel. Here, the "Connect" and "Disconnect" buttons are used to connect and disconnect, respectively, through the port, the communication between MATLAB and Arduino and, therefore, the user's communication with the device. Then, there are the buttons with numbered switch labels which have two things associated with them: a light and a graphic. This light represents the state of the LEDs of each switch, which will be green while the switch is "On" and will turn red once "Off".

Similarly, the graphs represent the digital value of the LEDs (0 if "Off" and 1 if "On") as a function of time from the start of the connection to the system.

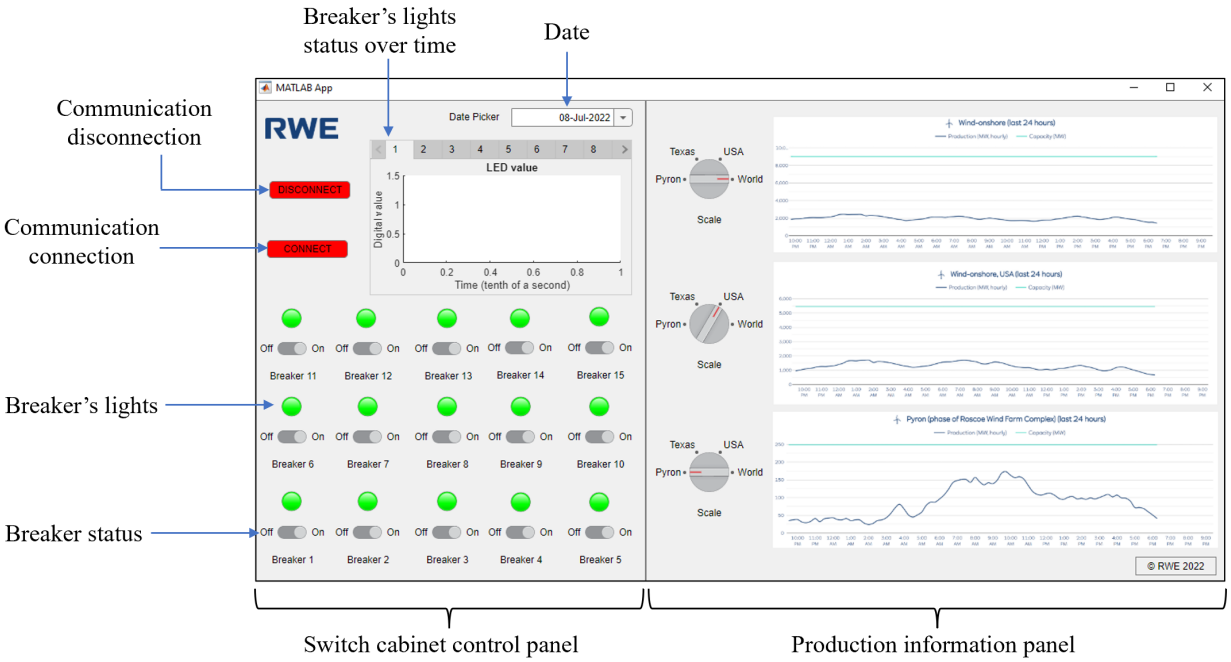


Figure 48. Graphical user interface initially

when the "Connect" button is pressed, the communication starts and the switch status graphs start to be displayed along the time it takes for a connection, programmed to last 30 seconds in this case. If all switches are on, their graphs will be giving a digital value of 1. This can be seen in *Figure 49.*

When a breaker is tripped, the system detects through the hall effect sensor the change of position and, instantly, it is shown on the interface display, both in the breaker status and its respective light and through the graph that will start reading the digital value 0, as shown in *Figure 50.*

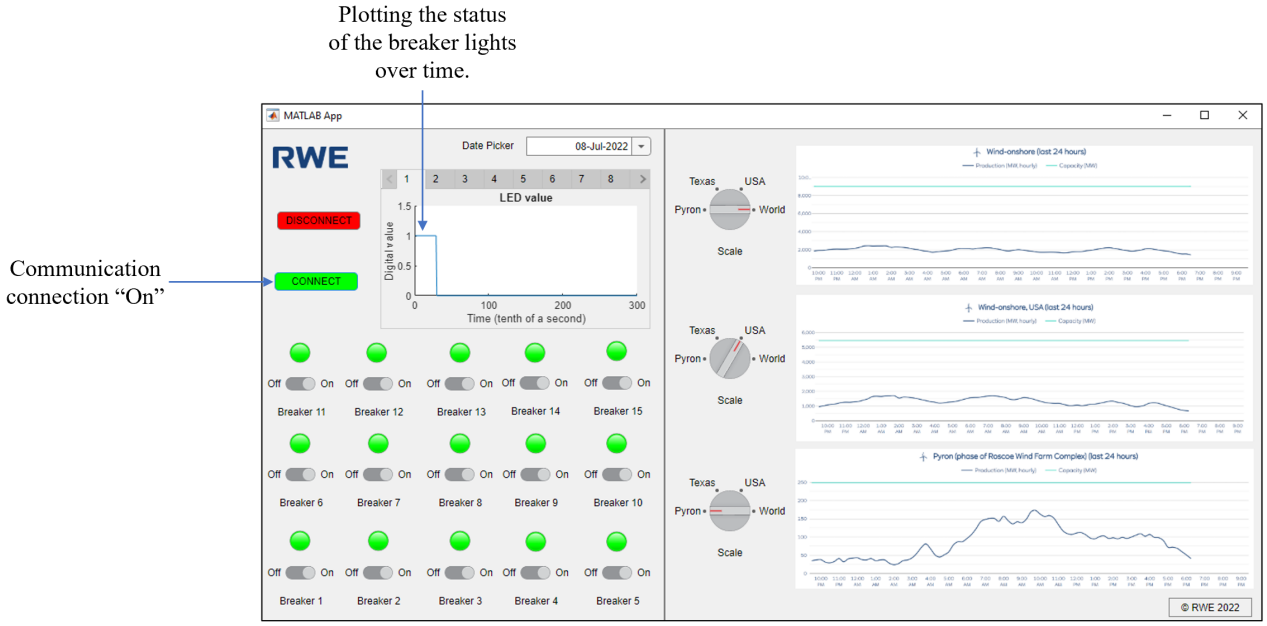


Figure 49. Graphical user interface after connection

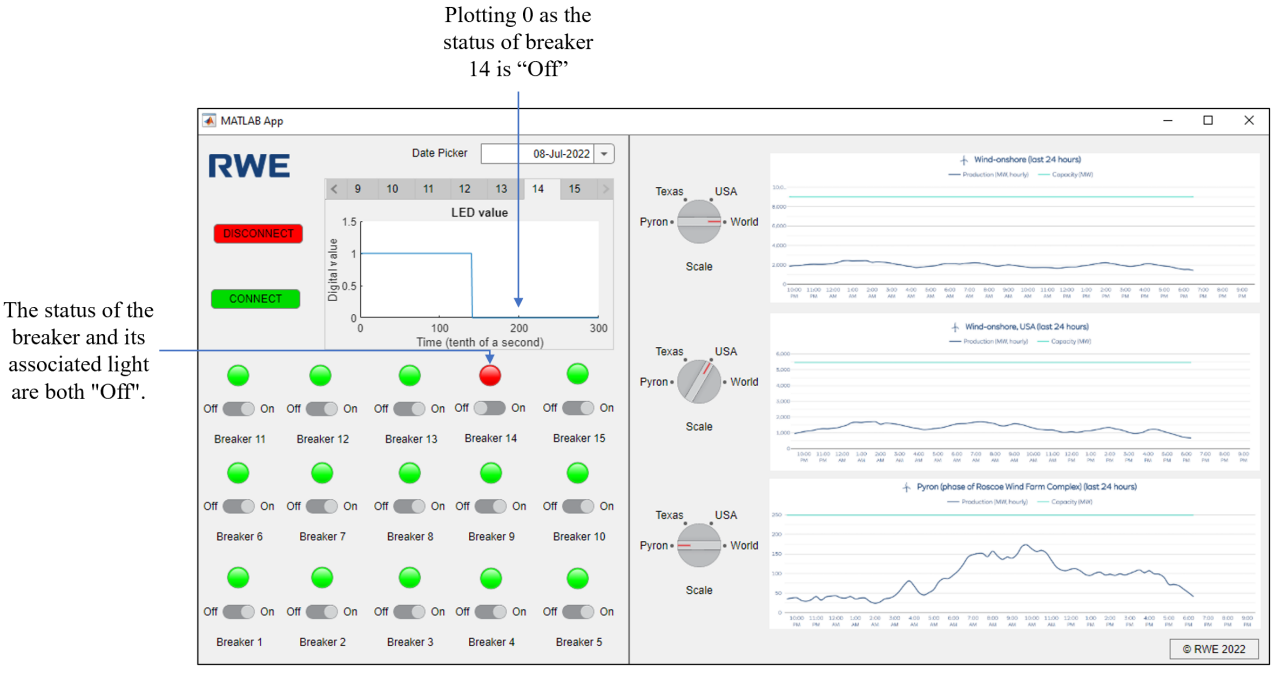


Figure 50. Circuit breaker 14 trips

When the user presses the reset button, the stepper motor would set the linear actuator to the exact position of the switch and the linear actuator would reset it, then return both to their

starting position, as it can be seen in *Figure 51*. Thus, when the on button is pressed again, the system and the linear actuator will be in their initial states *(Figure 49)*.

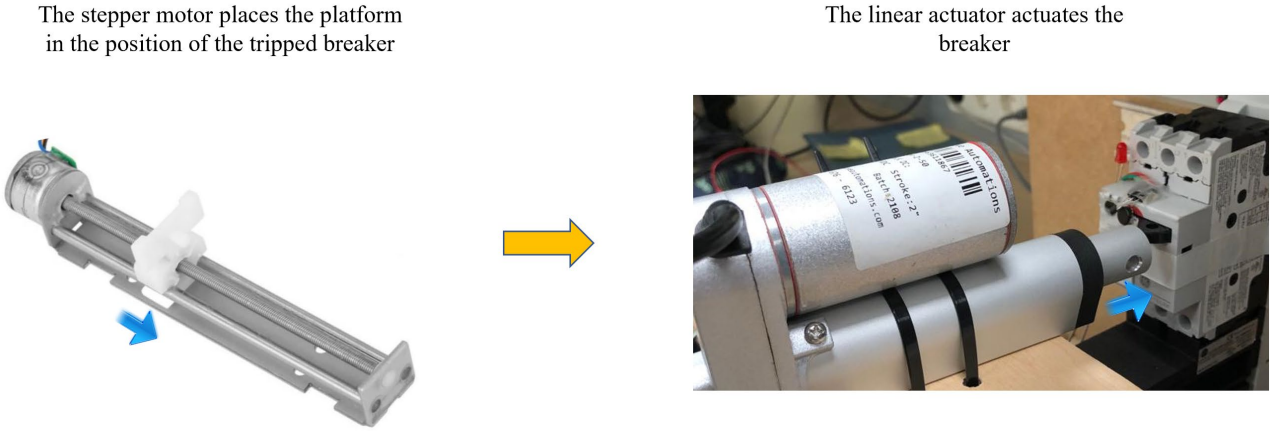


Figure 51. System resetting performance after user command

If due to any incident, such as excessive vibrations in the wind turbine, the BreakerBot does not actuate correctly to the switch, when reconnecting, the status of the switch will continue to appear as “Off”, so as not to generate false "correct resets" (*Figure 50*). The technician now decides whether to retest the reset or to go to the wind turbine.

Finally, when you want to terminate the communication, you can press the "Disconnect" button which will stop communicating with the port although it will remain defined in case you press "Connect" to resume the connection. *Figure 52* shows this scenario

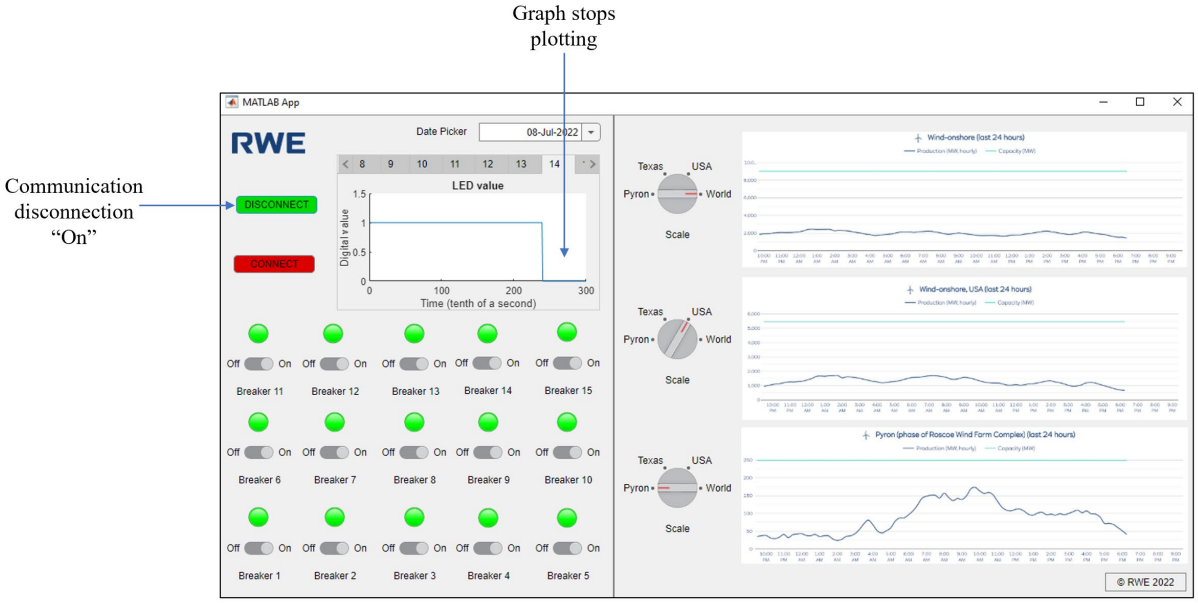


Figure 52. Graphical user interface after disconnection

4.4 OPC UA SERVER

The RWE systems to which information will be sent are Supervisory Control and Data Acquisition (SCADA) systems, which use hardware and software to improve system operation. Their objectives are to optimize automation and achieve faster decisions. Therefore, these systems require the use of OPC UA to enable the exchange of information between devices.

OPC UA is a protocol for implementing intra-machine, machine-to-machine, or machine-to-system communication. It enables the exchange of information while ensuring high data protection against threats and attacks. Specifically, its object structure allows it to be used to store, for example, measurement values, instrument characteristics, etc.

In this project, to implement the OPC UA architectures, the Raspberry PI board would have the function of making a server receive and transmit signals from sensors and actuators. These sensors and actuators are connected, as discussed in the *Chapter 3*, to an Arduino board which, being an open-source board like the Raspberry Pi, complies with this type of architecture and OPC services in the industrial area.

The proposed OPC UA system receives information and can control the behavior of sensors and actuators connected to an Arduino, through a server-client communication. The server is mounted on the Raspberry Pi to establish communication with the client through the internet. *Figure 53* shows the proposed OPC UA system.

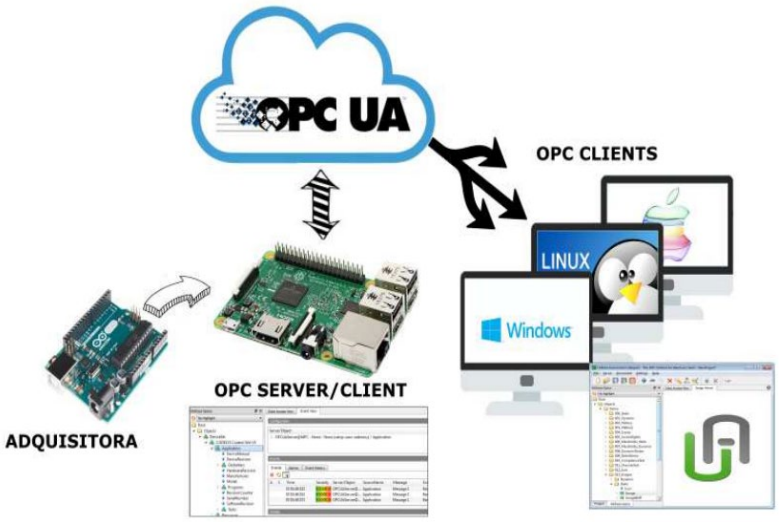


Figure 53. Proposed OPC UA system [38]

The implementation of this system, which would involve a parallel study, is summarized below. First, the initial conditions of the environment should be established by configuring the OPC UA server mounted on the Raspberry Pi, using certain libraries such as FreeOPCUa. Then, Arduino would need to be connected to the Raspberry Pi for the exchange of information about sensors and actuators. Finally, two Python codes have to be written, one to run the server and the other the client.

Chapter 5. TESTING

To test BreakerBot, RWE sent some sample breakers that were identical to the breakers used in their wind turbines. These samples are used to emulate the real-world conditions BreakerBot will be working in. Starting by testing the simplest components first, such as the sensors and motors. Doing so guarantees that any problems that appear during the building phase will be a consequence of the interaction of subcomponents instead of the subcomponents themselves. Once the subcomponents are tested, they will be integrated into the end product. The test plan consists of two separate phases. The first phase is local testing and the second is onsite testing. The goal is to do as much local testing as possible to reduce the likelihood of any major onsite failures. This means exhaustive test coverage and as much emulation of onsite conditions as possible.

5.1 BREAKER DETECTION TESTING

To test BreakerBot's breaker detection, the Hall-effect sensor will be wired into the detection microcontroller and monitor its outputs when the breaker is flipped vs. when it is unflipped. Once the sensor readings are calibrated and they are able to reliably detect a flipped breaker, the design will be moved onto multiple adjacent breakers. Here, it will be tested that a sensor won't be triggered when a breaker adjacent to it flips, ensuring that each breaker detection module is independent of adjacent modules. To test this, the sensor was connected to the Arduino and attached it to the top of the breaker over the left side of the switch while the permanent magnet was attached to left side of the breaker, to not interfere in the actuation. This is the position planned for the sensor to be in for the solution. Then, the sensor data was monitored using a serial monitor as the breaker was manually flipped back and forth.

The results of this test were that new magnets had to be purchased. The sensor worked but it needed to be too close to the magnet to produce reliable effects. After purchasing stronger rare-earth magnets, the detection system was retested using the same setup. The results from this test were much more favorable as it could clearly differentiate a tripped breaker vs an untripped breaker. Pictures from this test are below in *Figure 54.*

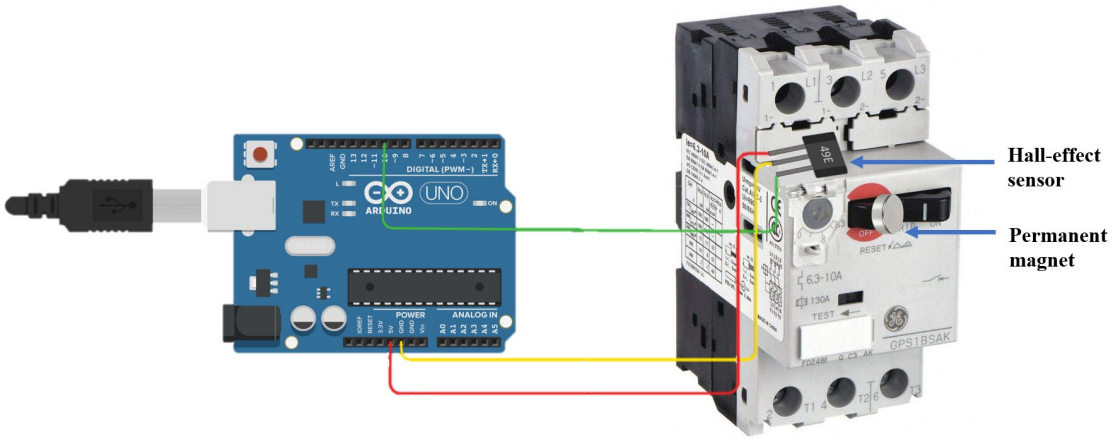


Figure 54. Breaker detection system

5.2 BREAKER ACTUATION TESTING

The main goal of the breaker actuation testing was to ensure that the actuator could produce enough force to actuate the breaker. Therefore, the first step towards actuation testing is independently measuring how much force is required to flip the breaker. If no accurate, reliable method to gather this data is found, breakers spec sheet will be used as a reference. The next step will be ensuring the actuators can be toggled on and off from our microcontroller, regardless of force output. To carry out actuation testing two independent parts are studied: linear actuator and stepper motor testing.

5.2.1 LINEAR ACTUATOR TESTING

Before testing the linear actuator on the breaker, it was necessary to first make sure that the actuator is functioning. To do this, the actuator was wired to a 12v single channel relay, connected to and driven by the Arduino. According to outside resources consulted while testing the actuator, sending a high voltage signal should've caused the actuator to extend, and a low voltage would make it retract. However, three hours of experimenting later there were no results as it wouldn't get to work. Further independent research revealed that the wrong kind of relay was picked. The problem was thinking a 12v single channel relay was needed because the actuator runs at 12v and there is only a single device attached to each relay. However, 5v dual channel relay was actually needed. 5v and 12v refers to the input voltage of the relay, not the output as thought. The dual channel was needed because a

single channel can only produce 0v and 5v, when the actuator needs -5v and +5v. Once using the correct relay channel, controlling the actuator was trivial.

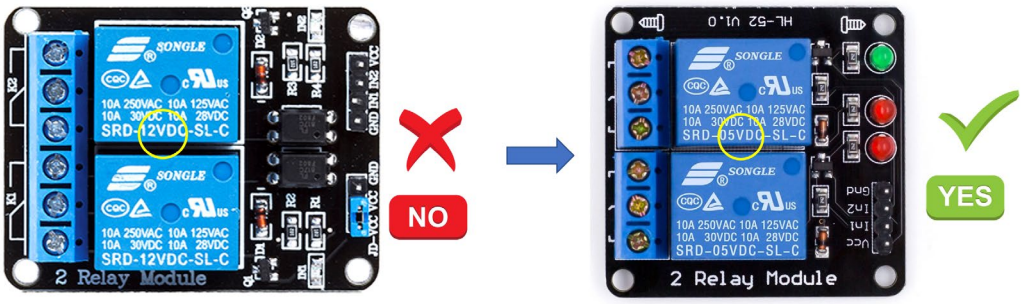


Figure 55. 12V vs 5V dual channel relay

The next step will be installing one of the actuators onto a sample breaker and test that it generates enough force to reliably flip the breaker. If it can, our testing is done. If it cannot, we will increase the force until it can. The linear actuator bought had enough force to flip the breaker, as calculated in *Section 3.3.2.2.*

To diminish the effect of counter forces, the linear actuator was mounted and breaker onto custom made clamp platforms. Doing this ensured both the actuator and breaker would be stable and supported, like they would be in the wind turbine. The actuator was lined in front of a breaker at a 45° angle so that it would contact the breaker switch head on while it was in a tripped state. An example is presented in *Figure 56.*

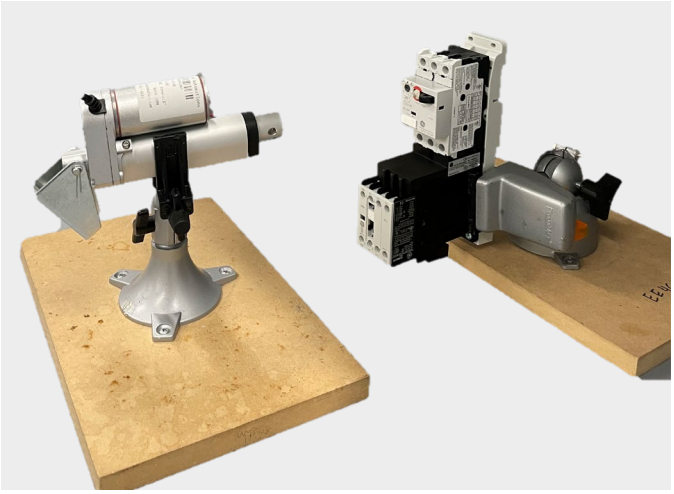


Figure 56. Initial actuation testing

Firstly, it was thought to have an incident angle would be essential to the actuation design because if the actuator was directly in front of the breaker, more force would be required. To test, the actuator was connected to the power supply to fully extend. Results were positive for this test as the actuator was more than capable of flipping the switch. It made it look so easy in fact that the test was done again, this time with the actuator directly facing the switch. Again, the actuator was able to flip the switch back on the breaker.

5.2.2 STEPPER MOTOR TESTING

Once the linear actuator had been tested, a similar procedure was carried out to test the operation of the stepper motor. The only challenge once the system was assembled was to set the particular direction and number of steps for each breaker position, but this can be easily changed in Arduino. Once it reaches the desired position, the linear actuator would actuate while the stepper motor platform waits for it to finish resetting and return to the home position. *Figure 57* a diagram of the stepper motor pitch and direction calibration.

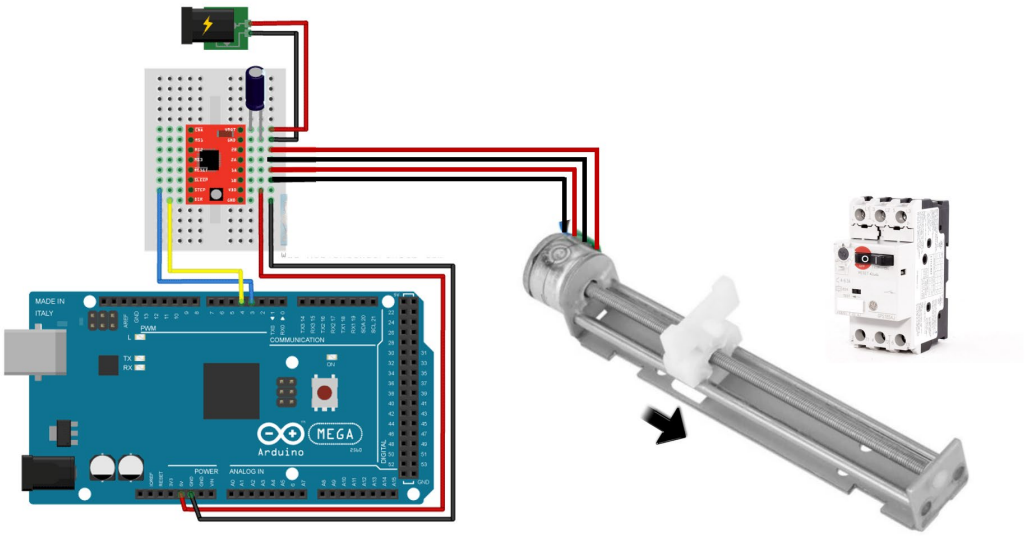


Figure 57. Stepper motor adjustment

5.3 COMMUNICATION TESTING

For the development of communication with both the system and the user, the process was divided into three stages: Arduino-system communication, Arduino-interface communication, and joint system communication.

In the first stage, Arduino-system communication, all the above tests were performed with individual codes to ensure that all elements of both detection and actuation responded separately. Once their response was verified, they were adjusted so that their behavior within the system was as desired. For example, the sensor and magnet were placed at the necessary distance so that the change in flow was perceived. Likewise, in the case of the linear actuator, the time for the motor to rotate clockwise and counterclockwise was adjusted, as well as the time for the motor to stop. For the stepper motor, both the number of pulses to achieve the turns that would take the platform to the target position and the frequency of these pulses that would be reflected in the speed of the motor were adjusted.

In the second stage, prior to deployment, Arduino-interface communication modules would be tested with simple tests.

In this way, an interface was created that simply consisted of a switch and a light from which the state of the LED placed on the Arduino board could be controlled. Thus, this code, developed in MATLAB, would be improved to ensure proper communication between Arduino and MATLAB in such a way that it could read the state of the circuit and act on it once the user decided to do so.

Finally, both communications should be merged into a single communication. To this end, the detection and actuation codes were combined into a single code that in turn communicated with MATLAB. It was necessary to organize the order and timing in which each element would come into operation so that the system would work as desired. In addition, a more complex interface capable of changing the states of a wind turbine control panel was developed, as shown in *Chapter 4*.

By performing many small tests as components get built, it was ensured to be working on a solid foundation and it was more likely to finish in time with no hidden bugs. A technician app would be deployed ahead of the onsite visit to ensure there won't be any problems and to focus on control system communication testing.

5.4 INTEGRATED SYSTEM TESTING

After separately testing the breaker detection system and our breaker actuation system, both will be combined for testing together. First of all, the frame of the BreakerBot will be built, respecting the dimensions of the cabinet, to orient the breaker and the system vertically, like how it will be installed inside the cabinet. Initially, a single device of the system will be tested onto a single breaker. If the system can properly detect a breaker flip, then reliably flip it back, the testing will be done. If it cannot, the interactions of our systems will be fine-tuned based on the specific problems it is facing to solve them. After the system works on an individual breaker, testing on the system on multiple adjacent breakers will be done to ensure there isn't interference between any two modules. When installed on site, testing will be necessary to ensure that it works under actual operating conditions. This may involve forcing a breaker to flip or waiting until one does.

5.4.1 INTEGRATION AND FABRICATION

Once the prototype was designed, the next step was to build the the structure to test the integration of all systems together. Before building the frame, a 1:1 replica of the cabinet was built, that can be found uptower out of plywood as can be seen in *Figure 58* below.

The cabinet

The wooden replica

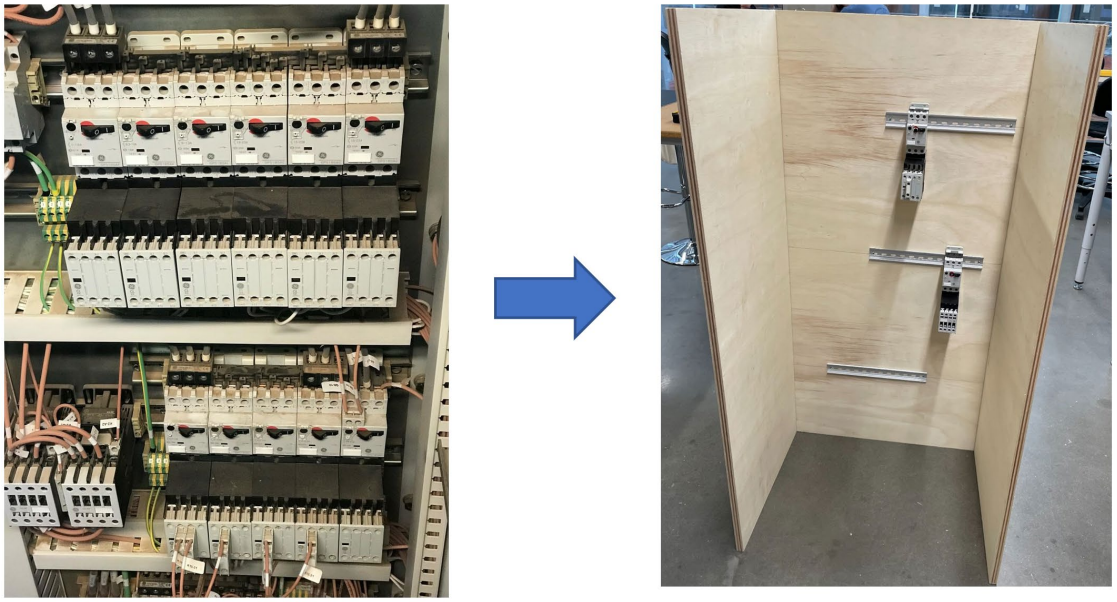


Figure 58. Original mock breaker cabinet and replica

This cabinet gave insight on the physical dimensions to work with. It allowed to see if the dimensions of BreakerBot would fit into the breaker cabinet in the turbine without having to physically take it there. It was also possible to mount 35mm DIN railing, which is the same railing used in the turbine. Doing this let us hang the breakers RWE sent, making the test environment even more similar to the production environment.

The first implementation of the prototype, designed to mount the components, was a 3-level shelving unit made out of wood. *Figure 59* below shows the result.

Figure 59. Wooden mounting frame

This structure was built with shelves to hold the linear actuators for each row of breakers. Each level on the structure was placed at the exact height of the corresponding row of breakers as given by RWE technicians. However, the weight of the wood, as well as its flammability and the fact that the height of the wood could not be adjusted, were major inconveniences in maintaining this structure. RWE informed that this solution was not viable for BreakerBot because of these disadvantages and proposed to research for metal solutions. This gave rise to the final 80/20 alum T-slot structure design. This frame was much more suited for the project because it was lighter, adjustable, fireproof and was closer to the prototype designed.

After finished constructing this frame, a single actuator was mounted onto the structure to test the detection-actuation loop. The assembly was carried out with 25 mm circuit hangers together with T-slot nuts. The frame was placed within the mock breaker cabinet and orientated it so that the mounted actuator was directly facing the actuator. A picture of this test is shown below in *Figure 60*. Then, the breaker was manually tripped and observed the controller detect the change, activate the linear actuator automatically and reset the breaker. The results of this test were a success. BreakerBot was successful in detecting and resetting a tripped breaker. It is important to note that this loop was cut off from any communication

with the technician web portal as all detection and actuation signals were sent and received locally from the Arduino.

Figure 60. Single actuator mounted onto the metal frame

Once the system was proven to work for one breaker, the device was reproduced to work for all breakers. Although in the final design only one linear actuator was placed in each of the rows, in the first delivery of the model, built at the University of Texas at Austin, each breaker was actuated with a particular linear actuator as can be seen in *Figure 61*.

Figure 61. Aluminum Frame

5.4.2 ON-SITE TESTING

In situ testing on one of the wind turbines under study is important and should be performed prior to using BreakerBot in order to carry out a series of simple but necessary adjustments for its correct operation. The onsite testing includes several tasks that are mentioned below:

1. Place BreakerBot inside the cockpit.
2. Adjust the device with stops so that it fits perfectly in the cockpit.
3. Adjust the number of steps the stepper motor has to make in each row to reach the switches.
4. Test the operation of the program and the software-hardware communication under wind turbine vibrations.
5. Check that the device can send and receive information to and from the control center.

6. Check the operation of the touch sensor system when closing the control cabinet.
7. Study of the feasibility of connecting the device to RWE's network.
8. Other inconveniences or disadvantages about the wind turbine environment that may occur during the visit.

Chapter 6. FINAL DESIGN

Once testing has been carried out and verified that it works correctly, the structure of the final design is presented, which will be made available to RWE for use in its wind turbines.

Initially, in the first BreakerBot design, linear actuators were installed at each switch position to be reset. However, subsequently, after stepper motor testing, the design was further improved to include rails in each row to bring the linear actuators to the set position of the breaker to be reset. *Figure 62* shows the prototype designed for the implementation of the final structure. It consists of three linear actuators each mounted on a platform that would move horizontally until the desired switch is reached and acted upon. The structure is hollow inside to lighten the weight, with the base slightly reinforced.

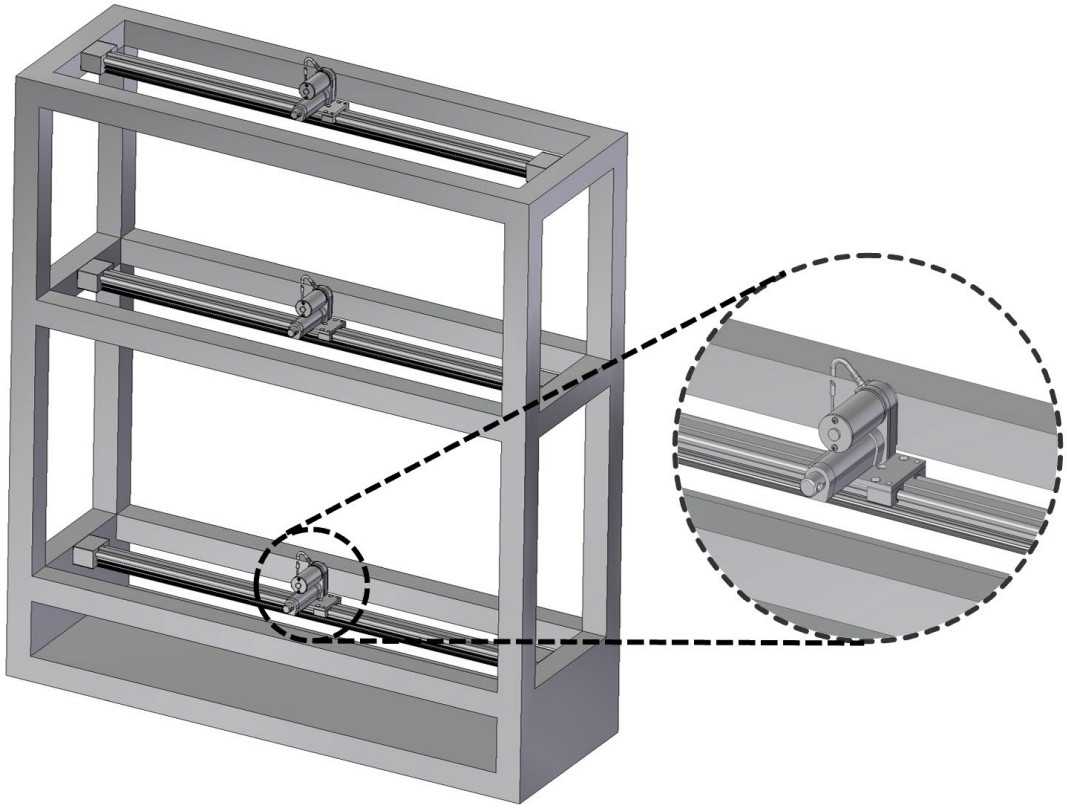


Figure 62. Final frame design of BreakerBot

6.1 DESIGN BLUEPRINT

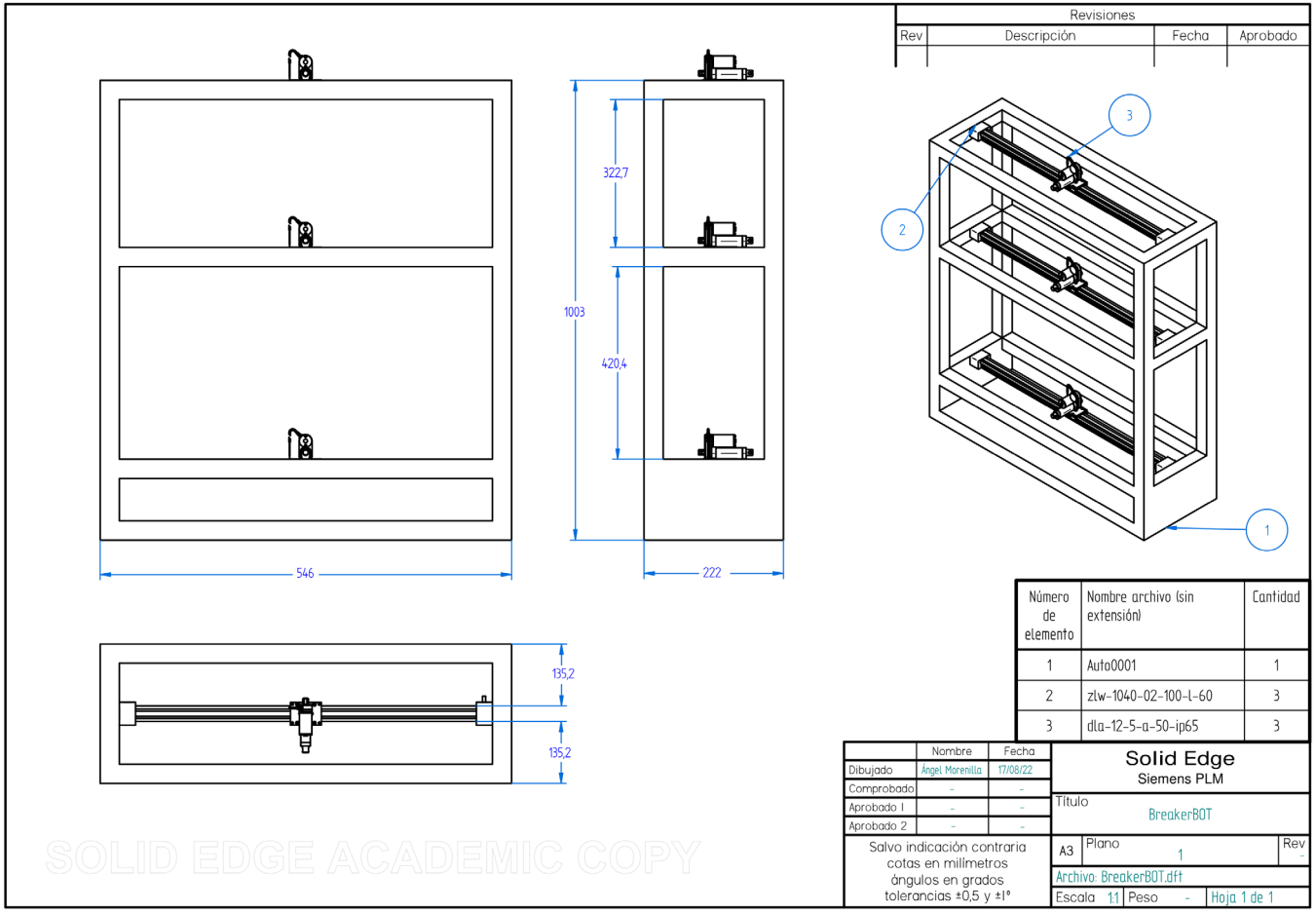


Figure 63. Design blueprint

The design plan is shown in *Figure 63*, where its dimensions can be observed. These dimensions are within the constraints of the cabinet (1219mm H x 584mm W x 381mm D), leaving a certain margin of tolerance in case it becomes necessary.

6.2 OPERATION

The entire operation of the system consists of the linkage of the detection, actuation, and communication system with both the technician and the company's system.

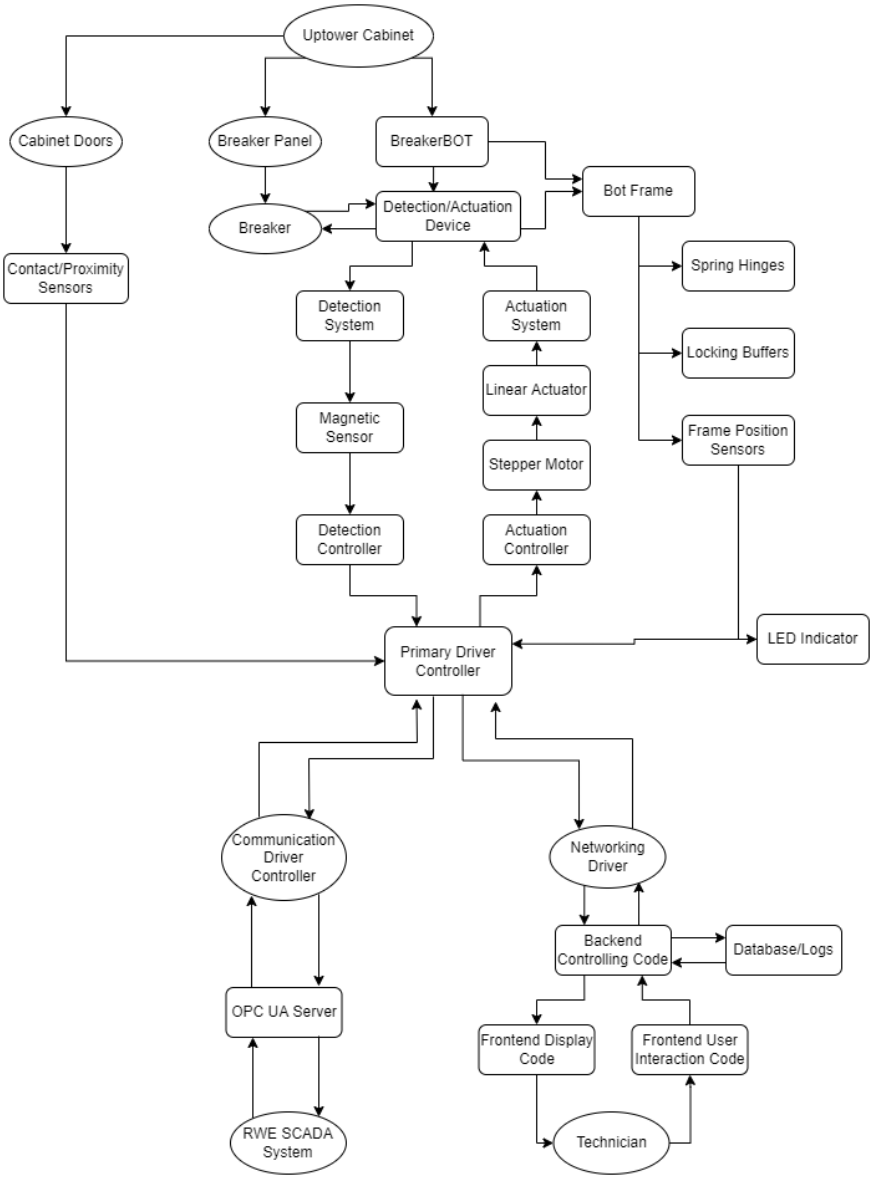


Figure 64. System operation

Continuing to follow the block diagram from *Figure 64*, the detection system is constantly reading information from its sensors. This information is then received and processed by the primary controller which communicates with both the networking driver and the communication driver controller. Once the networking controller receives the information and it reaches the database from the tower, the technician portal will display breaker status to the technician, providing the graphical user interface *(Figure 65)* to monitor and reset breakers. This command then follows back down through the same networking and communication components back to the primary controller, which will trigger the linear actuator and reset the flipped breaker.

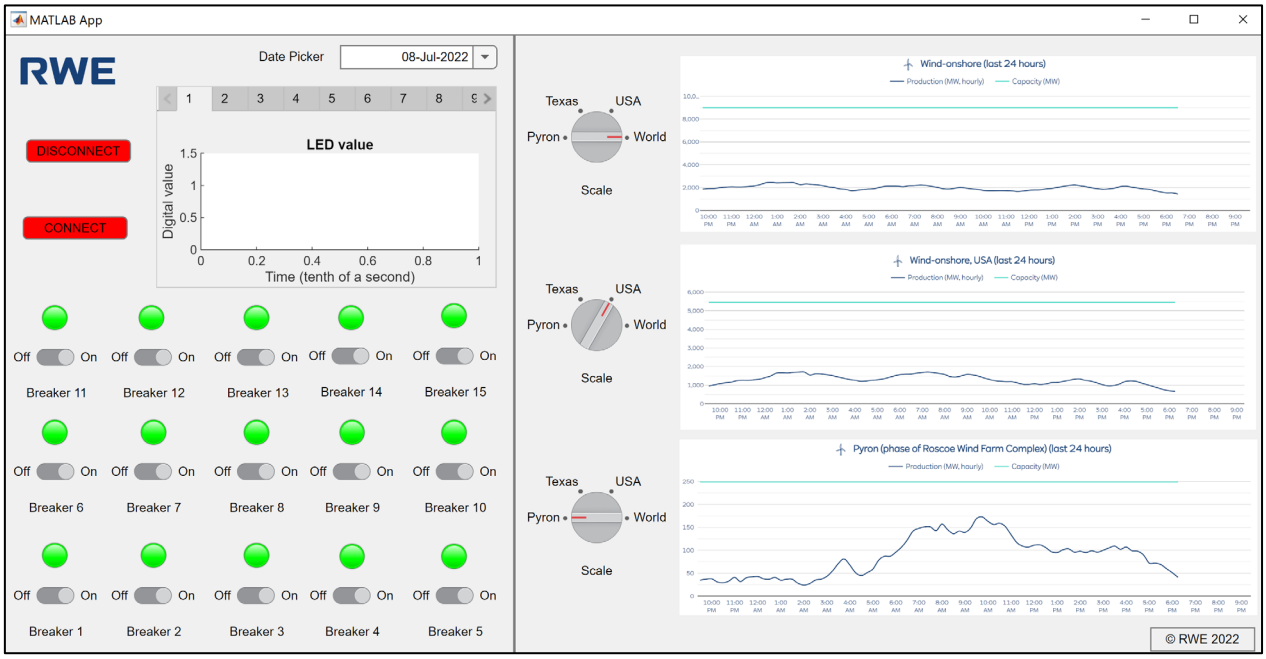


Figure 65. Interface design

The block diagram from *Figure 64* also outlines a minor subcomponent stemming from the risk reduction activities: a sensor to monitor the cabinet door. The design calls for a magnetic contact sensor to see if the cabinet door has been opened, if it has the system will go into an idle state where the system will not attempt to reset a switch regardless of an input. This is for technician safety to eliminate any chance for the actuation system to injure someone working in the cabinet, even if the system's movement is relatively small. Note that while in this state, the detection system will still be active as its stationary nature presents no harm to the technician.

6.3 IMPLEMENTATION

After the individual testing of the components and the design of the final metallic structure, everything would be implemented in the final design of the structure, as can be seen in the schematic in *Figure 66*. Its operation is as simple as putting all the previous tests together in an organized order:

1) Initially, the LED associated to each breaker will be "On" if everything is correct.
2) Arduino constantly reads inputs in the loop and sends them to MATLAB
3) If a breaker is tripped, the detection system will cause the sensor to notice the magnetic field change and the breaker's LED will turn off.
4) MATLAB's interface will show the user that the breaker has turned off.
5) If the user decides to reset it, MATLAB will send a signal to the Arduino to start the actuation system.
6) First, the stepper motor will drive the linear actuator to the switch position to be reset.
7) Then, the linear actuator will reset the breaker.
8) Finally, the stepper motor will return to its initial position.
9) the system is as it was at the beginning and will continue to read in a loop what happens in the control cabinet.

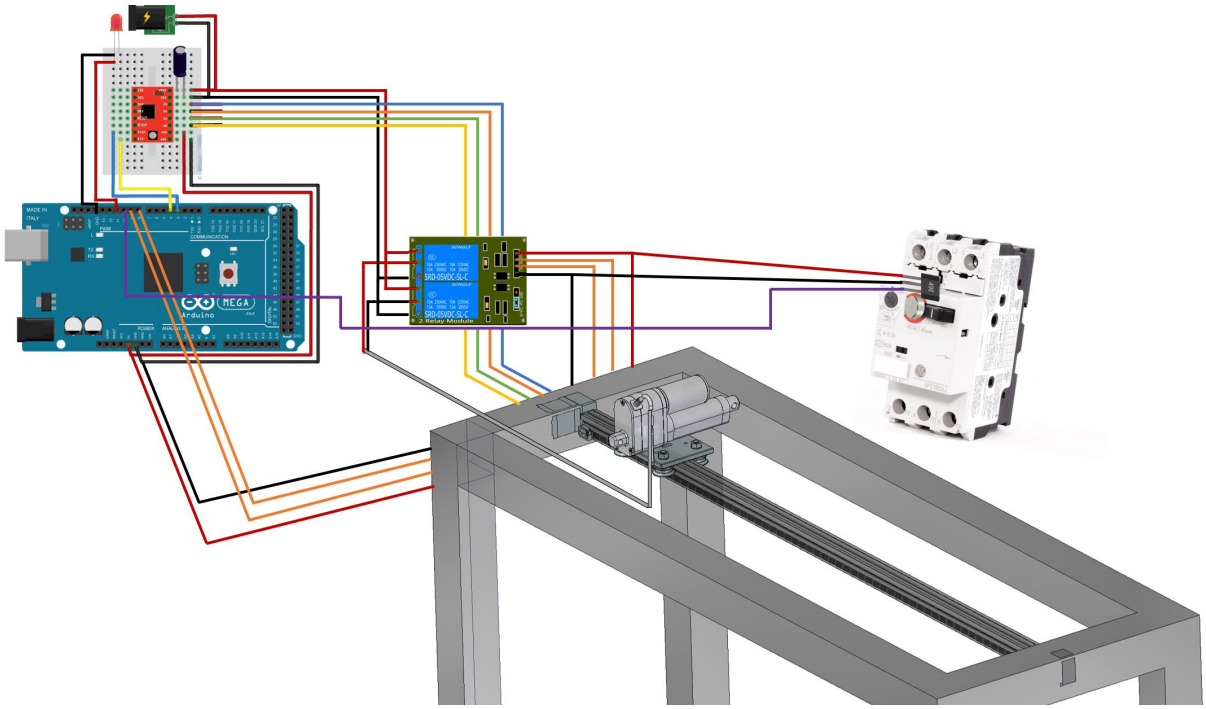


Figure 66.Final design implementation

The mounting of the final structure of BreakerBot is crafted using 80/20 1-inch T-slot aluminum extrusion. The structure is best described as an aluminum six level shelving unit. The structure is 1003 mm H x 546 mm W x 222 mm D. Each connected piece of t-slot extrusion is attached using 90° profile connectors along with 20-series t-sliding nuts and M5x8 mm hex screws. A 3 mm short arm allen wrench is required to tighten or loosen components attached using this combination of nuts and screws. On the 1st, 3rd, and 5th levels from the top are where the actuators are mounted. Each actuator will be mounted on the platform with two points of contact. The first is the front of the actuator, which is the side closer to the breakers. This was accomplished with 1 inch conduit hangers. The back side of the actuators, further from the breakers, are secured using ½ inch double sided conduit clamps.

On the 2nd, 4th, and 6th levels are thin aluminum plates, which were optional for the design but an appropriate place to place the electronics such as channel relays and Arduinos to adhere to. Every level where actuation is mounted has a level below it to attach associated electronics.

Finally, a terminal block was attached to the metal frame. They were added to the model as they enhance safety by grounding, isolating, and protecting other components in the circuit. Terminal blocks have finger-proof connections to prevent electric shock. In addition, terminal blocks can also provide test points, which add even more safety to the circuit. The terminal block mounted was divided in 4 rows and 2 columns, as seen in *Figure 67*. The columns of the left were to give supply to both the motors and the components connected to Arduino. The columns of the right is where all the electronic components for detection and actuation were placed.

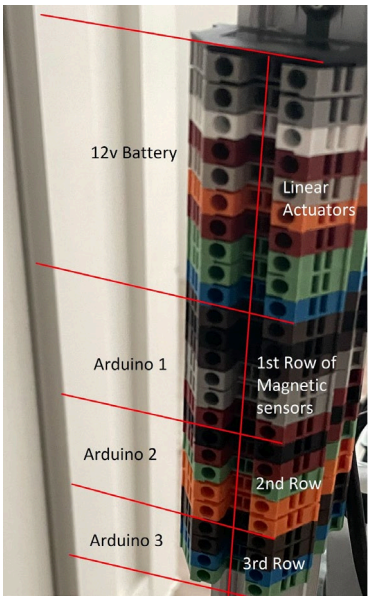


Figure 67. Terminal block diagram (Left-Supply, Right-Usage)

6.4 DESIGN CONSIDERATIONS

BreakerBot design proposed in this paper is intended for the laboratory study of the device as well as the onsite study in the wind turbine nacelle. Although it integrates communication via Bluetooth through which we can control the device remotely, this is only for remote testing. Once BreakerBot is ready to be reproduced and used in more turbines, RWE should integrate the transmission of the information provided by the device to the nearest control center. There, the program I have developed should be installed on the computers so that either an operator or a technician can control the behavior of the tripped breakers.

In fact, the form of remote communication with the device is a question that the company should decide whether to communicate remotely via the SCADA itself (it would be the best) or not to "disturb" the SCADA via conventional communication network. Sometimes the company has its own network (PLC with signaling or similar), and if not the conventional one via router.

Another point to take into account is the power supply of BreakerBot. Both the company and wind turbine experts guarantee that the turbine nacelle has a power supply close to 12V. However, in case the situation arises, when testing on site, that this power supply is not available, an alternative to power BreakerBot would be to use a diode bridge rectifier.

The diode bridge rectifier's objective is to convert an AC input to a DC output. Bridge rectifiers include four diodes in a configuration that makes the output voltage to have the same polarity for either polarity of input voltage. The purpose of adding a capacitor to this circuit is to reduce the ripple of the output voltage to get closer to a constant value.

Thus, a variac would be connected to the power grid and by means of the relationship shown in *Figure 68*, the desired DC voltage would be obtained.

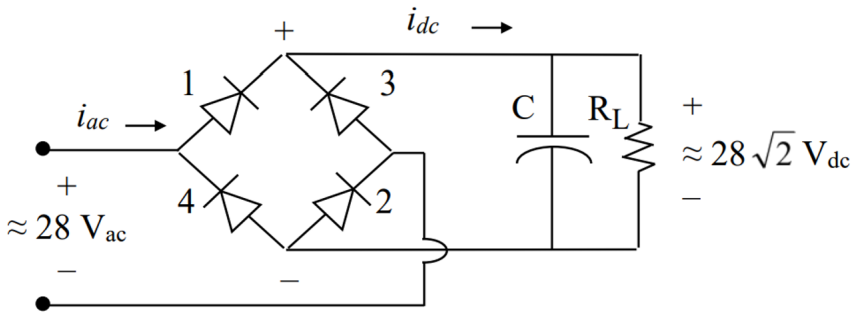


Figure 68. Diode Bridge Rectifier Circuit [50]

Chapter 7. CONCLUSIONS

7.1 CONCLUSIONS

During the development of the project, the following objectives have been satisfactorily met:

1. Optimized design of the BreakerBot within the cabin dimensions.
2. Design flexible and scalable to be deployed across multiple turbine types and thousands of units.
3. Design installable and operable on top of existing OEM hardware.
4. Integration of visual feedback of breakers, contacts and diagnostic lights up tower.
5. Guarantee to maintain the safety chain of the existing turbine.
6. Assembly and physical integration of all hardware elements.
7. Development of the BreakerBot control software:
    a. Communication between detection of the tripped breaker and technician.
    b. Communication between technician response and actuation.
    c. Remote accessibility.
    d. Design to store data in a data base.
    e. Design to communicate securely via NERC CIP.
8. Control software integration of all sensors and actuators.
9. Design to ensure risk reduction:
    a. Correct reinstallation.
    b. Technician safety and automatic shutoff.
    c. Operating temperature.
    d. Fragility and strength of material.

Overall, it can be stated that, although BreakerBot still has some on-site testing to be done, it is a big step forward in finding the best solution for RWE. It has managed to achieve its objectives in the simplest possible way, so that the structure can be easily understood by anyone and adapted to existing wind turbine systems. The circuit breaker actuation system will reliably return the circuit breakers back into place on command, as many tests have been

performed on the detection and actuation part of the circuit breakers. In addition, the way everything has been physically implemented in one structure allows us to easily adjust to different wind turbine designs and technicians can change the position of the linear actuators when necessary or even the layout of the entire structure. As a result, it allows technicians to access the cabinet with ease, which was initially one of the main requirements set by our company RWE. For the future, there is still work to be done, such as finishing the development of the communication component and the integration of multiple Arduino controllers. Once these challenges are met, BreakerBot can be further developed, being an opportunity to follow many interesting and unique paths. This could be extension to other types of turbines or, for example, custom built circuit boards to relieve stress on the controllers, or a joint project between RWE and its turbine manufacturers. BreakerBot may continue to evolve over the coming semesters.

7.2 RECOMMENDATIONS

With the project in its current state, there are several recommendations for a production quality version of the system, as well as recommendations on alternative approaches to the problem. This system is only a prototype and that a production version will likely go through some sort of evolution and given the experience with the system we have ideas on how to clean up the system for production and simplify any further development. In addition, the approach has been constrained by limited time and resources, and different ways of approaching this same problem are proposed under different circumstances.

7.2.1 IMMEDIATE ACTION ITEMS

To continue with the development of this prototype, the first step should be to complete the communication component. Within this component, the most important task remaining is to test the interconnection between the primary controller and the communication controller. This is the major bottleneck to completing the rest of the communication system, as there is no way to actually test this component until data flows through that connection. The next step would be to complete the software on the Raspberry Pi to send the current states of the switches to the web portal server and via OPC to the RWE SCADA system. Without knowing how the Raspberry Pi would receive that information,

that software could not have been written previously. Once the communication is complete, only minor integration/fabrication tasks remain: the adjustable feet would need to be installed on the rack suspenders so that the space between them and the cabinet wall could be minimized. Finally, the system would only need to be installed on site and tested.

7.2.2 REFINEMENTS FOR A PRODUCTION QUALITY SYSTEM

The first recommendation is to create a custom PCB for each pair of sensors and actuators to hold the logic, circuitry, and relay required. One of the problems faced is that each one of these pairs takes up too many pins on the controller, requiring to build out more controllers than it was believed necessary. A custom PCB could reduce the number of pins required for each breaker (potentially down to one depending on the design), reducing the number of controllers required for the entire system. This would clean up the system as a whole by reducing the number of cables throughout the system and simplify the software required for the primary controller as it would have to interact and account for fewer secondary controllers. Following this, it is recommended that the system is organized so that each row or set (of a predefined number) of breakers is grouped together and controlled by a single secondary controller. This again would simplify the code in the primary controller, as it would know exactly how much data it will receive from each secondary controller, as well as further cleanup and simplify the overall solution.

7.2.3 ALTERNATIVE DESIGN

It was also contemplated the idea of having a single actuator that would be attached to a system that could maneuver it in the X-Y dimension to wherever the tripped breaker was. This would have been a more interesting design that would probably also work but this decision had to be made earlier. We decided not to do so due to our lack of experience with this type of system, but for the future development of this project it may be a viable option.

7.2.4 OTHER APPROACHES TO THE PROBLEM

While this system was built specifically for RWE, it is a fair assumption that other wind turbine operators would want a similar solution, thus I recommend that anyone working on

this problem also work with the wind turbine manufacturer to build their system as an OEM or preinstalled system. By doing so, you would likely be afforded more flexibility in your design since the entire cabinet could be adjusted if needed. The primary benefit is you would be able to integrate your communication into the turbine's existing monitoring system, both simplifying your development of the communication system and simplifying the operator's monitoring of the turbines since they would no longer have to switch between systems. Along the same lines, there would also be benefits in working with the circuit breaker manufacturers to integrate this system into the circuit breakers themselves. Remote monitoring and resetting have already been accomplished in smaller household circuit breakers, and those ideas can be extended into the larger commercial-grade breakers used in these turbines. The design would be able to monitor the internal mechanisms of the breakers, reducing the overall footprint of the system and providing more accurate readings, and could integrate resetting mechanisms internally, further reducing footprint.

7.3 DESIGN CONTEMPLATIONS AND CHANGES

This subsection describes the changes and modifications made to the design during implementation, as well as the main problems encountered.

7.3.1 DESIGN CHANGES

The prototype built, as well as the one presented in the final design, is drastically different from the one that was originally proposed. The ideas for actuation were scrapped after the breaker changed. As the design changed from a rack and pinion to just a linear actuator. At the time, it was planned to have every actuator/sensor combination tucked into a 3D printed box that would be supported by the breaker itself. However, after handling the new linear actuators, it was learned that it wouldn’t be possible to support them solely from the breakers. The actuators were too large and too heavy. Instead, a way to mount them independently of the breakers was needed. This led to weeks of discussions with RWE about what the possibilities were, what the restrictions were, and what we were capable of within the timeframe. Then, eventually, we were pointed to 80/20® T-slot aluminum

extrusion. This was the perfect solution as it was lightweight, workable, standardized, and fire-proof.

7.3.2 SAFETY AND ETHICAL ASPECTS OF DESIGN

BreakerBot was designed with safety in mind. With every component made of metal or other non-flammable materials, BreakerBot poses no additional safety concerns to technicians while they are in the wind turbine. And since turbines must be off while a technician provides maintenance, there are no risks that BreakerBot would accidentally tamper with a live turbine while a technician is uptower. In fact, BreakerBot increases workplace safety by requiring fewer tower climbs from technicians. Each time BreakerBot successfully rearms a breaker, that is one less time a technician has to climb up and down the tower, which is a risky endeavor.

7.3.3 CHALLENGES ALONG THE WAY

As in many other projects, unforeseen challenges arose that threw the project off course. The drive design had to be redone after a communication error within RWE meant that the intended circuit breakers were not the same as those that would operate in the actual turbines. The photos of a tower cabinet that were used as the basis for the design showed switches with a vertical lever, very similar to those in a domestic electrical cabinet, and all designs were based on this premise. However, the switches delivered for testing in mid-February, and which can be seen throughout this report, had a horizontal lever much more similar to a button in terms of operation. Therefore, redesign of the switch sensing and actuation systems had to be carried out. This did not take too long in itself, but it did cause ripple effects in parts sourcing. Detection and actuation parts previously ordered had to be returned and new parts had to be researched, selected and ordered. With the current supply chain delays, these new parts took longer to arrive, further delaying progress. We did not have enough time to thoroughly work through some of the low-level technical aspects of the new design, which resulted in facing those aspects for the first time during implementation, further slowing down development and ultimately causing us to miss our final deadlines.

However, this wasn't altogether the worst thing because it actually simplified the actuation design a lot. In the end, it was possible to successfully pivot to a new performance design, but it set the project behind. Then, there were some problems in the development and installation of the software that delayed the development schedule of the technical portal by a week. Finally, after several days of debugging and research, a program was found that fit the needs. Also, there were some issues with delayed shipments for the aluminum extrusion, which was nerve-wracking as this was already late into the project cycle. To resolve this, the group worked overtime when the shipment finally came to get our project off the ground.

7.4 ECONOMIC ANALYSIS

So far, the project has cost RWE around $2500 worth of supplies and materials for testing and implementing. RWE reports that each turbine gets two "false alarm" breaker trips per week and each one takes about 2 hours for a technician to reach the site, diagnose the problem, reset the breaker, then return. The cost of this 2-hour downtime is the cost of the technician's time that could be spent on other important work + the cost of lost energy during this downtime. RWE has not disclosed their technician salary to us but taking the average in Texas of $55,000 per year, it is estimated that 2 hours of salary to be $55. A windmill like the ones we are working with would produce about 500 kWh during those 2 hours of downtime. That energy would cost about $42.35 at 8.37¢ per kWh, the current average in Texas for commercial power. Upon successful installation, the system would save RWE about $98 per week per turbine on average. This means that if RWE were to only install BreakerBot into one turbine, RWE will have made a profit from this project in 7 months. If they decide to expand it to another 10 turbines similar to the one under study, RWE will have made 100x profit in 5 years. This is a very positive result for BreakerBot.

7.5 THE SUSTAINABLE DEVELOPMENT GOALS

The Sustainable Development Goals (SDG) aim to evolve towards a better and more sustainable future. To this end, they seek to reduce global problems by encouraging the inclusion of habits in companies and people in general so that a transition to a more socially, economically and environmentally sustainable world can take place.

The study and development of BreakerBot contributes to the fulfillment and achievement of 3 of the 17 Global Goals adopted by the United Nations Organization (UNO): Affordable and Clean Energy; Industry, Innovation and Infrastructure; Responsible Consumption and Production.

7.5.1 GOAL 7- AFFORDABLE AND CLEAN ENERGY

In recent years, there has been an increase in the number of people with access to electricity, however, along with the exponential growth of the population will come an increase in the demand for energy, and the high dependence on fossil fuels could have repercussions on the climate of planet Earth. Thus, it is necessary to invest and believe in renewable energies, capable of providing clean energy without leaving a footprint on our planet.

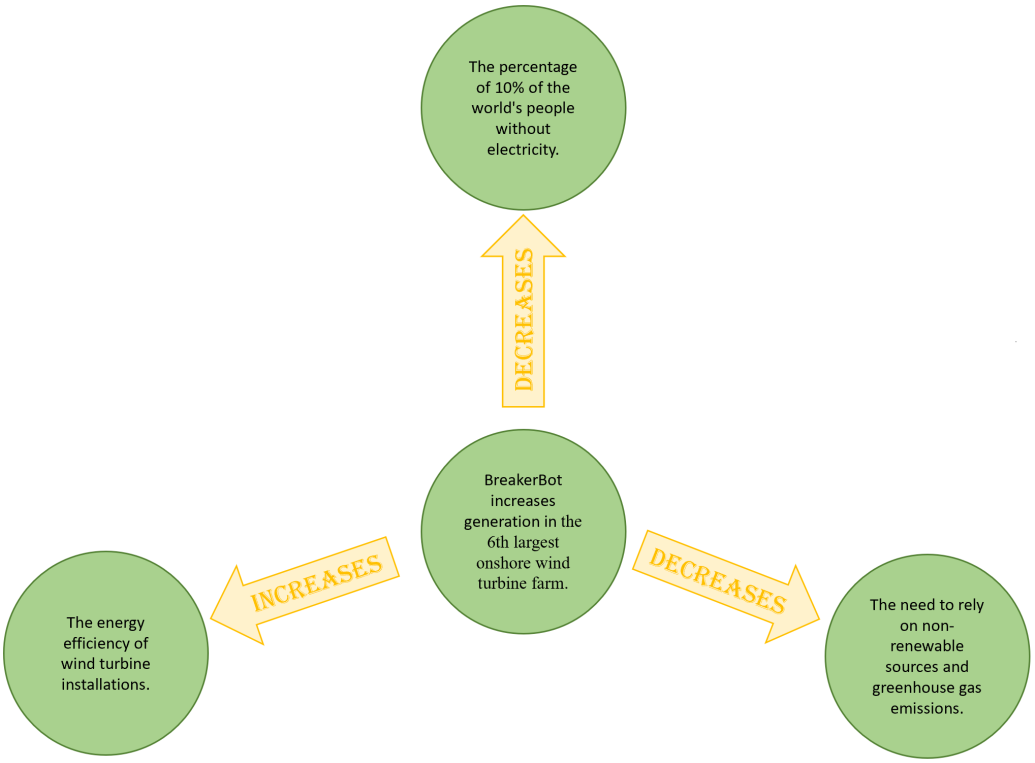


Figure 69. Impact of BreakerBot over GOAL 7

As BreakerBot's goal is to increase renewable energy generation through increased wind turbine uptime, this project has a major impact on obtaining clean energy at the sixth

largest onshore wind turbine farm in the world as of 2022, The Roscoe Wind Farm. The diagram in *Figure 68* discusses some of the objectives to which BreakerBot contributes.

7.5.2 GOAL 9- INDUSTRY, INNOVATION AND INFRASTRUCTURE

Objective 9, which focuses on investment in infrastructure and the role of renewable energies, is in line with Objective 7. It also advocates technological advances as a solution to economic and environmental challenges. Furthermore, it is committed to the creation of new jobs and the promotion of energy efficiency. *Figure 69* shows how BreakerBot is involved in improving the industry and infrastructure.

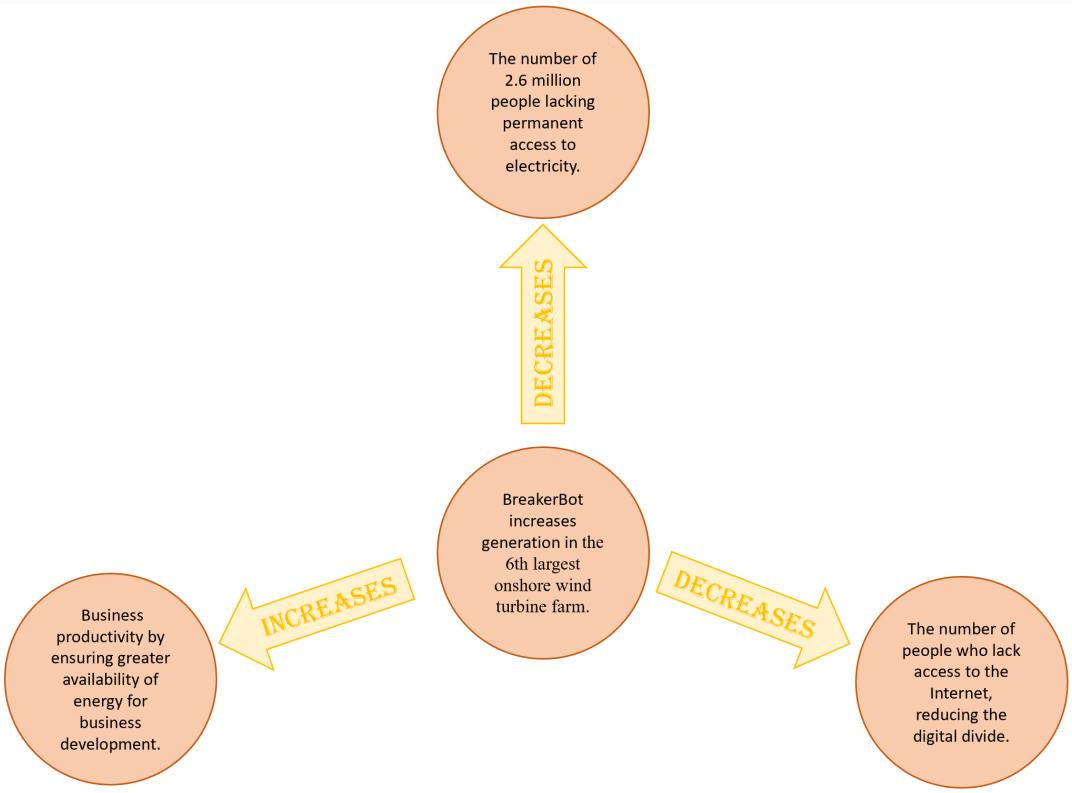


Figure 70. Impact of BreakerBot over GOAL 9

7.5.3 GOAL 12- RESPONSIBLE CONSUMPTION AND PRODUCTION

In order to achieve sustainable development, it is necessary to reduce the ecological footprint by changing production and resource consumption methods. Good management of shared natural resources and the reduction of toxic and polluting waste play a fundamental role in this objective. The contribution made by BreakerBot to achieve Objective 12 can be found in the resolution of the problems shown in *Figure 70*.

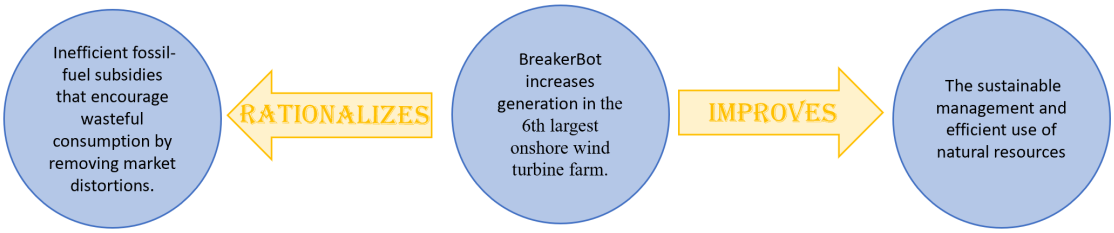


Figure 71. Impact of BreakerBot over GOAL 12

Chapter 8. BIBLIOGRAPHY AND REFERENCES

[11] Library of Congress – *Renewable Energy Industries: A Research Guide*

[12] SwitchBot – *SwitchBot Bot*

[13] A. Ogawa *et al.*, *"Tendon-driven Elastic Telescopic Arm - Integration of Linear Motion and Bending Motion"*, *2020 IEEE/SICE International Symposium on System Integration (SII)*, 2020, pp. 1328-1334.

[14] Q. R. Xu *et al.*, *"Stick-On Piezoelectromagnetic AC Current Monitoring of Circuit Breaker Panels"*, in *IEEE Sensors Journal*, vol. 13, no. 3, pp. 1055-1064, March 2013.

[15] F. Sun, W. Liu and J. Fan, *"Application of video image recognition technology in substation equipments monitoring", 2011 International Conference on Electrical and Control Engineering*, 2011, pp. 4374-4377.

[16] William B. Cushman, *"Telescoping robot arm with spherical joints"*, U.S. Patent 5 410 994, May 2, 1995.

[17] Xianging Liu *et al., "Device for serving live power lines"*, U.S. Patent 9 972 981 B2, May 15, 2018.

[18] Michael L. Agronin *et al.*, *"Remote controlled wall switch actuator"*, U.S. Patent 7 608 793 B2, October 17, 2009.

[19] Russell Thomas Watford, "Circuit breaker trip notification systems and methods", U.S. Patent 9 054 516 B2, June 9, 2015.

[20] Arduino – *What is Arduino?*

[21] eTechnophiles – *Introduction to ATmega328p Pinout, datasheet and specifications*

[22] NicePNG – *Placa Arduino Uno Transparent*

[23] Solectro – *2 Channel Relay Module DC 12V 10A Low/High Trigger for Arduino*

[24] Quora – *How do you control a DC motor in both directions with two relays (motor, relay, Arduino)?*

[25] Instructables circuits – *DC Motor Controller With Two Relay*

[26] Bernhard Jakoby, Isaku Kanno and Loes Segerink, *"Sensors and Actuators A: Physical"*, Elsevier, 2000

[27] J. R. Brauer, *"Hall Effect and Magnetoresistive Sensors"*, IEEE, 2006

[28] Movil Tronics – *Sensor Efecto Hall 49E*

[29] EEEProject – *Touch Sensor Working Principle And Application*

[30] Aoran Xu, Liu Zhang, Cailian Gu and YiZhang, *"Crowbar circuit in wind power grid low voltage ride through"*, 2014 China International Conference on Electricity Distribution (CICED), 2014

[31] ElectronicsNotes – *SCR Thyristor Crowbar: overvoltage protection circuit*

[32] Stephen D Umans, *"Fitzgerald & Kingsley's Electric Machinery"*, McGraw-Hill Education, 2014

[33] Power Technology – *The Roscoe Wind Farm Project, Texas, USA*

[34] Ronergy – *Top 10 largest wind farms in the world*

[35] U.S Energy Information Administration – *Net generation: Pyron Wind Farm LLC Hybrid (56981) : wind : all primemovers : annual*

[36] U.S Energy Information Administration – *Champion Wind Farm LLC, monthly*

[37] U.S Energy Information Administration – *Roscoe Wind Farm LLC, monthly*

[38] U.S Energy Information Administration – *Inadale Wind Farm LLC, monthly*

[39] Statista – *RWE's electricity generation in 2020 and 2021*

[40] RWE – *Investor's presentation*

[41] Braden Houston, "Technical Description and Specifications Wind Turbine Generator System GE Wind Energy 1.5SLE 60 Hz", Noble Environmental Power, 2005

[42] Synacorp Technologies – *Super strong round disc neodymium magnet*

[43] Daraz – *12V Stepper Motor Stroke 80Mm With Slider Motor*

[44] DroneBot – *Stepper Motors with Arduino*

[45] Motion Control – *Stepper motor or servomotor: Which should it be?*

[46] Progressive Automations – *PA-14 Data Sheet*

[47] AranaCorp – *Arduino and Bluetooth module HC-06*

[48] F. Rivera-Velazquez, E. Salazar-Valle and G. M. Martínez-Águilar, *"OPC UA server on Raspberry Pi and Arduino for didactic use", 2021 10th International Conference On Software Process Improvement (CIMPS)*, 2021, pp. 115-124

[49] Program de las Naciones Unidad para el Desarrollo – *Objetivos de Desarrollo Sostenible*

[50] M. Flynn, "Diode Bridge Rectifier", The University of Texas at Austin, Austin, TX, EE 462L: Power Electronics Laboratory course, Spring 2022, page 2.

Appendix I. Budget

The budget developed in this appendix covers the cost of the hardware and software elements and tools used for the study, design and implementation of BreakerBot. Obviously, some purchased elements such as, for example, the stepper motor are not used in the final design due to its size and motor strength, but they are used to simulate its control and operation within the system.

The project easily met the budget constraints set by RWE, but although deadlines were consistently met at the beginning of the project, unforeseen changes in specifications and delays in development caused a delay in the later stages of the project. RWE provided us with a total budget of $10,000, which would cover all necessary parts and travel to the site. From the final bill of materials below, the total cost was $2,915.31, well under budget. This total cost does not include travel to the site, as it was not possible to test the system on an actual turbine, but it is reasonable to say that travel to the site would not have put us over budget. The site is a short drive from Austin and lodging there is reasonably priced (approximately $150-200 per night), so it is unreasonable for travel to the site to cost more than $7,000.

Hardware/ Software	Item	Quantity	Purpose	Cost
Hardware	Conduit Clamps	25	Mount actuators to frame	$50.61
Hardware	80/20 Aluminum Extrusion	16x 800mm	Aluminum frame of BreakerBot	$166.98
Hardware	DIN Rail Terminal Block Kit	2	Provide stable points for wiring actuators to power sources	$58.98
Hardware	Aluminum Profile Connector Set	4	To connect the 80/20 aluminum together	$115.96
Hardware	5v 2 Channel Relay Module	15	Provides appropriate 5v and -5v source for actuator	$53.97
Hardware	12x12x1/16 Aluminum Sheet Metal	3	Serve as shelves for electronic equipment onboard BreakerBot	$47.97
Hardware	Gikfun A3144 Hall Effect Sensor	15	Used to detect magnet attached to breaker switch	$8.28
Hardware	35 mm DIN Rail	4	Used in mock breaker cabinet to hang breakers. Also used to mount terminal blocks alongside BreakerBot	$19.98
Hardware	NAZZO rare earth magnets	15	Adhered to breaker switch for trip detection	$16.19
Hardware	2" Linear Actuators	15	Used to rearm breakers	$1,471.50
Hardware	4' x 2' x .75" Plywood	5	Used to build mock breaker cabinet	$196.00
Hardware	Arduino Uno REV3	15	Primary detection/actuation controller	$341.55
Hardware	Raspberry Pi 4 Model B	1	Communication Controller	$173.90
Hardware	Gorilla Glue mounting putty	1	Adhere various components	$8.40
Hardware	12v Rechargable Battery Pack	1	Powers system	$21.99
Hardware	D8-motor80 Dc 12v Stepper Motor	1	Used to position the linear actuator	$16.80
Software	MATLAB and Simulink Students Suite	1	To develop the graphical interface	$146.25
			TOTAL	$2,915.31

Appendix II. Arduino script

```
int LED= 12;
int SENSOR= 10;
int act_value= 0;
int relay1 = 8;
int relay2 = 9;
const int stepPin = 3;
const int dirPin = 4;


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Communication speed
  Serial.setTimeout(10);

  pinMode(LED, OUTPUT); //set pin as output for LED
  pinMode(SENSOR, INPUT); //set pin as input for sensor
  pinMode(relay1, OUTPUT); //set pin as output for relay 1
  pinMode(relay2, OUTPUT); //set pin as output for relay 2
  pinMode(stepPin,OUTPUT); //set pin as output for number of steps
  pinMode(dirPin,OUTPUT); //set pin as output for direction
  digitalWrite(LED, HIGH); //initialize the value of the LED as
HIGH

}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED, digitalRead(SENSOR));
  Serial.println(digitalRead(LED));

  while (digitalRead(SENSOR)==0){
      if(Serial.available()){ //Asking is it is possible to read
from serialport
        act_value= Serial.read(); //If possible, start reading
        if (act_value=='a'){
        digitalWrite(dirPin,HIGH); // Enables the motor to move
away from motor
        // Makes 200 pulses for making one full cycle rotation
        for(int x = 0; x < 1000; x++) {
          digitalWrite(stepPin,HIGH);
          delayMicroseconds(600);
          digitalWrite(stepPin,LOW);
          delayMicroseconds(600);
        }
        delay(2000);
```

```
         // Rotate in CCW direction
        digitalWrite(relay1, LOW);// turn relay 1 ON
        digitalWrite(relay2, HIGH);// turn relay 2 OFF
        delay(1000);// wait for 3 seconds

       // stop the motor
        digitalWrite(relay1, HIGH);// turn relay 1 OFF
        digitalWrite(relay2, HIGH);// turn relay 2 OFF
        delay(2000);// stop for 2 seconds

       // Rotate in CW direction
        digitalWrite(relay1, HIGH);// turn relay 1 OFF
        digitalWrite(relay2, LOW);// turn relay 2 ON
        delay(1000);// wait for 3 seconds

       // stop the motor
        digitalWrite(relay1, HIGH);// turn relay 1 OFF
        digitalWrite(relay2, HIGH);// turn relay 2 OFF
        delay(2000);// stop for 2 seconds

        digitalWrite(dirPin,LOW); //Enables the motor to move
towards from motor
        // Makes 400 pulses for making two full cycle rotation
        for(int x = 0; x < 1000; x++) {
          digitalWrite(stepPin,HIGH);
          delayMicroseconds(600);
          digitalWrite(stepPin,LOW);
          delayMicroseconds(600);
        }
         Serial.println(digitalRead(SENSOR));
      }

      }
    }
      delay(100);
   }
```

Appendix III. MATLAB script

```
classdef finalapp < matlab.apps.AppBase

% Properties that correspond to app components
properties (Access = public)
UIFigure matlab.ui.Figure
GridLayout matlab.ui.container.GridLayout
LeftPanel matlab.ui.container.Panel
TabGroup matlab.ui.container.TabGroup
Tab matlab.ui.container.Tab
UIAxes_16 matlab.ui.control.UIAxes
Tab_2 matlab.ui.container.Tab
UIAxes_2 matlab.ui.control.UIAxes
Tab_3 matlab.ui.container.Tab
UIAxes_17 matlab.ui.control.UIAxes
Tab_4 matlab.ui.container.Tab
UIAxes_18 matlab.ui.control.UIAxes
Tab_5 matlab.ui.container.Tab
UIAxes_19 matlab.ui.control.UIAxes
Tab_6 matlab.ui.container.Tab
UIAxes_20 matlab.ui.control.UIAxes
Tab_7 matlab.ui.container.Tab
UIAxes_21 matlab.ui.control.UIAxes
Tab_8 matlab.ui.container.Tab
UIAxes_22 matlab.ui.control.UIAxes
Tab_9 matlab.ui.container.Tab
UIAxes_23 matlab.ui.control.UIAxes
Tab_10 matlab.ui.container.Tab
UIAxes_24 matlab.ui.control.UIAxes
Tab_11 matlab.ui.container.Tab
UIAxes_25 matlab.ui.control.UIAxes
Tab_12 matlab.ui.container.Tab
UIAxes_26 matlab.ui.control.UIAxes
Tab_13 matlab.ui.container.Tab
UIAxes_27 matlab.ui.control.UIAxes
Tab_14 matlab.ui.container.Tab
graph matlab.ui.control.UIAxes
Tab_15 matlab.ui.container.Tab
UIAxes_29 matlab.ui.control.UIAxes
DatePicker matlab.ui.control.DatePicker
DatePickerLabel matlab.ui.control.Label
Connect matlab.ui.control.StateButton
Disconnect matlab.ui.control.StateButton
Breaker15Switch matlab.ui.control.Switch
Breaker15SwitchLabel matlab.ui.control.Label
Breaker14Switch matlab.ui.control.Switch
```

```
Breaker14SwitchLabel matlab.ui.control.Label
Breaker13Switch matlab.ui.control.Switch
Breaker13SwitchLabel matlab.ui.control.Label
Breaker12Switch_2 matlab.ui.control.Switch
Breaker12Switch_2Label matlab.ui.control.Label
Breaker11Switch_2 matlab.ui.control.Switch
Breaker11Switch_2Label matlab.ui.control.Label
Lamp_15 matlab.ui.control.Lamp
Lamp_14 matlab.ui.control.Lamp
Lamp_13 matlab.ui.control.Lamp
Lamp_12 matlab.ui.control.Lamp
Lamp_11 matlab.ui.control.Lamp
Breaker10Switch matlab.ui.control.Switch
Breaker10SwitchLabel matlab.ui.control.Label
Breaker9Switch matlab.ui.control.Switch
Breaker9SwitchLabel matlab.ui.control.Label
Breaker8Switch matlab.ui.control.Switch
Breaker8SwitchLabel matlab.ui.control.Label
Breaker7Switch matlab.ui.control.Switch
Breaker7SwitchLabel matlab.ui.control.Label
Breaker6Switch matlab.ui.control.Switch
Breaker6SwitchLabel matlab.ui.control.Label
Lamp_10 matlab.ui.control.Lamp
Lamp_9 matlab.ui.control.Lamp
Lamp_8 matlab.ui.control.Lamp
Lamp_7 matlab.ui.control.Lamp
Lamp_6 matlab.ui.control.Lamp
Breaker5Switch_2 matlab.ui.control.Switch
Breaker5Switch_2Label matlab.ui.control.Label
Breaker4Switch_2 matlab.ui.control.Switch
Breaker4Switch_2Label matlab.ui.control.Label
Lamp_5 matlab.ui.control.Lamp
Lamp_4 matlab.ui.control.Lamp
Breaker3Switch_2 matlab.ui.control.Switch
Breaker3Switch_2Label matlab.ui.control.Label
Lamp_3 matlab.ui.control.Lamp
Breaker2Switch_2 matlab.ui.control.Switch
Breaker2Switch_2Label matlab.ui.control.Label
Lamp_2 matlab.ui.control.Lamp
Lamp matlab.ui.control.Lamp
Breaker1Switch_2 matlab.ui.control.Switch
Breaker1Switch_2Label matlab.ui.control.Label
Image3 matlab.ui.control.Image
RightPanel matlab.ui.container.Panel
EditField matlab.ui.control.EditField
ScaleKnob_3 matlab.ui.control.DiscreteKnob
ScaleKnob_3Label matlab.ui.control.Label
ScaleKnob_2 matlab.ui.control.DiscreteKnob
ScaleKnob_2Label matlab.ui.control.Label
```

```
ScaleKnob matlab.ui.control.DiscreteKnob
ScaleKnobLabel matlab.ui.control.Label
Image4 matlab.ui.control.Image
Image2 matlab.ui.control.Image
Image matlab.ui.control.Image
end

% Properties that correspond to apps with auto-reflow
properties (Access = private)
onePanelWidth = 576;
end

properties (Access = private)
com % Description
end

% Callbacks that handle component events
methods (Access = private)

% Code that executes after component creation
function startupFcn(app)
delete(instrfind({'Port'},{'COM3'}));
%Port connection
app.com = serialport("COM3", 9600,"ByteOrder","little-endian");
%Connection warning
warning('on','MATLAB:serial:fscanf:unsuccesfulRead');
%Leds color is initialized
app.Connect.BackgroundColor='red';
app.Disconnect.BackgroundColor='red';
app.Lamp_14.Color='green';
app.Breaker14Switch.Value= 'On';


end

% Value changed function: Connect
function Connection(app, event)
%Connection with Arduino starts
app.Connect.BackgroundColor='green';
app.Disconnect.BackgroundColor='red';
fopen(app.com);
%Creation of matrix of 1 row and 200 colums to store data read
%from Arduino
%Graphic value is intialized
mat_ini=ones(1,300);
mat=zeros(1,300);
%The program is for switch_14, but will be the same code for
%the rest
for co=1 : 1 : 300
```

```
volt =str2double(fscanf(app.com, '%c'));
disp(volt);
if volt ~= 0
mat(co) = volt;
plot(app.graph, mat);
plot(app.UIAxes_2, mat);
plot(app.UIAxes_16, mat);
plot(app.UIAxes_17, mat);
plot(app.UIAxes_18, mat);
plot(app.UIAxes_19, mat);
plot(app.UIAxes_20, mat);
plot(app.UIAxes_21, mat);
plot(app.UIAxes_22, mat);
plot(app.UIAxes_23, mat);
plot(app.UIAxes_24, mat);
plot(app.UIAxes_25, mat);
plot(app.UIAxes_26, mat);
plot(app.UIAxes_27, mat);
plot(app.UIAxes_29, mat);
end
if volt== 0
%El interruptor se ha apagado
app.Lamp_14.Color= 'red';
app.Breaker14Switch.Value= 'Off';
break;
end

end
plot(app.UIAxes_2, mat_ini);
plot(app.UIAxes_16, mat_ini);
plot(app.UIAxes_17, mat_ini);
plot(app.UIAxes_18, mat_ini);
plot(app.UIAxes_19, mat_ini);
plot(app.UIAxes_20, mat_ini);
plot(app.UIAxes_21, mat_ini);
plot(app.UIAxes_22, mat_ini);
plot(app.UIAxes_23, mat_ini);
            plot(app.UIAxes_24, mat_ini);
            plot(app.UIAxes_25, mat_ini);
            plot(app.UIAxes_26, mat_ini);
            plot(app.UIAxes_27, mat_ini);
            plot(app.UIAxes_29, mat_ini);

            %Time to send a response to Arduino, can be changed
            %If time is over, just click connect again
        end

        % Value changed function: Disconnect
        function Disconnection(app, event)
```

```
            fclose(app.com);
            app.Connect.BackgroundColor='red';
            app.Disconnect.BackgroundColor='green';
        end

        % Value changed function: Breaker14Switch
        function ResetBreaker(app, event)
            % Compares if the user reset the button
            TF=  strcmp(app.Breaker14Switch.Value, 'On');
                if TF==1
                    app.Breaker14Switch.Value='On';
                    app.Lamp_14.Color='green';
                    fwrite(app.com, 'a');
                else
                    app.Breaker14Switch.Value='On';
                    app.Lamp_14.Color='green';
                end
            pause(20);
            app.Breaker14Switch.Value='On';
            app.Lamp_14.Color='green';
        end

        % Changes arrangement of the app based on UIFigure width
        function updateAppLayout(app, event)
            currentFigureWidth = app.UIFigure.Position(3);
            if(currentFigureWidth <= app.onePanelWidth)
                % Change to a 2x1 grid
                app.GridLayout.RowHeight = {596, 596};
                app.GridLayout.ColumnWidth = {'1x'};
                app.RightPanel.Layout.Row = 2;
                app.RightPanel.Layout.Column = 1;
            else
                % Change to a 1x2 grid
                app.GridLayout.RowHeight = {'1x'};
                app.GridLayout.ColumnWidth = {486, '1x'};
                app.RightPanel.Layout.Row = 1;
                app.RightPanel.Layout.Column = 2;
            end
        end

        % Value changed function: Breaker2Switch_2
        function Close(app, event)
            fclose(app.com);
            delete(app.com);
            delete(app);
        end
    end

    % Component initialization
```

```
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Get the file path for locating images
            pathToMLAPP = fileparts(mfilename('fullpath'));

            % Create UIFigure and hide until all components are
created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.AutoResizeChildren = 'off';
            app.UIFigure.Position = [100 100 1203 596];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.SizeChangedFcn = createCallbackFcn(app,
@updateAppLayout, true);

            % Create GridLayout
            app.GridLayout = uigridlayout(app.UIFigure);
            app.GridLayout.ColumnWidth = {486, '1x'};
            app.GridLayout.RowHeight = {'1x'};
            app.GridLayout.ColumnSpacing = 0;
            app.GridLayout.RowSpacing = 0;
            app.GridLayout.Padding = [0 0 0 0];
            app.GridLayout.Scrollable = 'on';

            % Create LeftPanel
            app.LeftPanel = uipanel(app.GridLayout);
            app.LeftPanel.Layout.Row = 1;
            app.LeftPanel.Layout.Column = 1;

            % Create Image3
            app.Image3 = uiimage(app.LeftPanel);
            app.Image3.Position = [16 536 90 50];
            app.Image3.ImageSource = fullfile(pathToMLAPP,
'RWE_Logo_2020.svg.png');

            % Create Breaker1Switch_2Label
            app.Breaker1Switch_2Label = uilabel(app.LeftPanel);
            app.Breaker1Switch_2Label.HorizontalAlignment =
'center';
            app.Breaker1Switch_2Label.Position = [21 17 58 22];
            app.Breaker1Switch_2Label.Text = 'Breaker 1';

            % Create Breaker1Switch_2
            app.Breaker1Switch_2 = uiswitch(app.LeftPanel,
'slider');
            app.Breaker1Switch_2.Position = [31 54 37 16];
app.Breaker1Switch_2.Value = 'On';
```

```
% Create Lamp
app.Lamp = uilamp(app.LeftPanel);
app.Lamp.Position = [36 88 25 25];

% Create Lamp_2
app.Lamp_2 = uilamp(app.LeftPanel);
app.Lamp_2.Position = [130 88 25 25];

% Create Breaker2Switch_2Label
app.Breaker2Switch_2Label = uilabel(app.LeftPanel);
app.Breaker2Switch_2Label.HorizontalAlignment = 'center';
app.Breaker2Switch_2Label.Position = [114 17 58 22];
app.Breaker2Switch_2Label.Text = 'Breaker 2';

% Create Breaker2Switch_2
app.Breaker2Switch_2 = uiswitch(app.LeftPanel, 'slider');
app.Breaker2Switch_2.ValueChangedFcn = createCallbackFcn(app,
@Close, true);
app.Breaker2Switch_2.Position = [124 54 37 16];
app.Breaker2Switch_2.Value = 'On';

% Create Lamp_3
app.Lamp_3 = uilamp(app.LeftPanel);
app.Lamp_3.Position = [228 88 25 25];

% Create Breaker3Switch_2Label
app.Breaker3Switch_2Label = uilabel(app.LeftPanel);
app.Breaker3Switch_2Label.HorizontalAlignment = 'center';
app.Breaker3Switch_2Label.Position = [212 17 58 22];
app.Breaker3Switch_2Label.Text = 'Breaker 3';

% Create Breaker3Switch_2
app.Breaker3Switch_2 = uiswitch(app.LeftPanel, 'slider');
app.Breaker3Switch_2.Position = [222 54 37 16];
app.Breaker3Switch_2.Value = 'On';

% Create Lamp_4
app.Lamp_4 = uilamp(app.LeftPanel);
app.Lamp_4.Position = [319 87 25 25];

% Create Lamp_5
app.Lamp_5 = uilamp(app.LeftPanel);
app.Lamp_5.Position = [416 87 25 25];

% Create Breaker4Switch_2Label
app.Breaker4Switch_2Label = uilabel(app.LeftPanel);
app.Breaker4Switch_2Label.HorizontalAlignment = 'center';
app.Breaker4Switch_2Label.Position = [303 17 58 22];
```

```
app.Breaker4Switch_2Label.Text = 'Breaker 4';

% Create Breaker4Switch_2
app.Breaker4Switch_2 = uiswitch(app.LeftPanel, 'slider');
app.Breaker4Switch_2.Position = [313 54 37 16];
app.Breaker4Switch_2.Value = 'On';

% Create Breaker5Switch_2Label
app.Breaker5Switch_2Label = uilabel(app.LeftPanel);
app.Breaker5Switch_2Label.HorizontalAlignment = 'center';
app.Breaker5Switch_2Label.Position = [401 17 58 22];
app.Breaker5Switch_2Label.Text = 'Breaker 5';

% Create Breaker5Switch_2
app.Breaker5Switch_2 = uiswitch(app.LeftPanel, 'slider');
app.Breaker5Switch_2.Position = [411 54 37 16];
app.Breaker5Switch_2.Value = 'On';

% Create Lamp_6
app.Lamp_6 = uilamp(app.LeftPanel);
app.Lamp_6.Position = [36 205 25 25];

% Create Lamp_7
app.Lamp_7 = uilamp(app.LeftPanel);
app.Lamp_7.Position = [130 205 25 25];

% Create Lamp_8
app.Lamp_8 = uilamp(app.LeftPanel);
app.Lamp_8.Position = [228 205 25 25];

% Create Lamp_9
app.Lamp_9 = uilamp(app.LeftPanel);
app.Lamp_9.Position = [319 205 25 25];

% Create Lamp_10
app.Lamp_10 = uilamp(app.LeftPanel);
app.Lamp_10.Position = [416 205 25 25];

% Create Breaker6SwitchLabel
app.Breaker6SwitchLabel = uilabel(app.LeftPanel);
app.Breaker6SwitchLabel.HorizontalAlignment = 'center';
app.Breaker6SwitchLabel.Position = [21 135 58 22];
app.Breaker6SwitchLabel.Text = 'Breaker 6';

% Create Breaker6Switch
app.Breaker6Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker6Switch.Position = [31 172 37 16];
app.Breaker6Switch.Value = 'On';
```

```
% Create Breaker7SwitchLabel
app.Breaker7SwitchLabel = uilabel(app.LeftPanel);
app.Breaker7SwitchLabel.HorizontalAlignment = 'center';
app.Breaker7SwitchLabel.Position = [114 135 58 22];
app.Breaker7SwitchLabel.Text = 'Breaker 7';

% Create Breaker7Switch
app.Breaker7Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker7Switch.Position = [124 172 37 16];
app.Breaker7Switch.Value = 'On';

% Create Breaker8SwitchLabel
app.Breaker8SwitchLabel = uilabel(app.LeftPanel);
app.Breaker8SwitchLabel.HorizontalAlignment = 'center';
app.Breaker8SwitchLabel.Position = [212 135 58 22];
app.Breaker8SwitchLabel.Text = 'Breaker 8';

% Create Breaker8Switch
app.Breaker8Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker8Switch.Position = [222 172 37 16];
app.Breaker8Switch.Value = 'On';

% Create Breaker9SwitchLabel
app.Breaker9SwitchLabel = uilabel(app.LeftPanel);
app.Breaker9SwitchLabel.HorizontalAlignment = 'center';
app.Breaker9SwitchLabel.Position = [303 135 58 22];
app.Breaker9SwitchLabel.Text = 'Breaker 9';

% Create Breaker9Switch
app.Breaker9Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker9Switch.Position = [313 172 37 16];
app.Breaker9Switch.Value = 'On';

% Create Breaker10SwitchLabel
app.Breaker10SwitchLabel = uilabel(app.LeftPanel);
app.Breaker10SwitchLabel.HorizontalAlignment = 'center';
app.Breaker10SwitchLabel.Position = [398 135 64 22];
app.Breaker10SwitchLabel.Text = 'Breaker 10';

% Create Breaker10Switch
app.Breaker10Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker10Switch.Position = [411 172 37 16];
app.Breaker10Switch.Value = 'On';

% Create Lamp_11
app.Lamp_11 = uilamp(app.LeftPanel);
app.Lamp_11.Position = [36 314 25 25];

% Create Lamp_12
```

```
app.Lamp_12 = uilamp(app.LeftPanel);
app.Lamp_12.Position = [128 314 25 25];

% Create Lamp_13
app.Lamp_13 = uilamp(app.LeftPanel);
app.Lamp_13.Position = [228 314 25 25];

% Create Lamp_14
app.Lamp_14 = uilamp(app.LeftPanel);
app.Lamp_14.Position = [318 314 25 25];

% Create Lamp_15
app.Lamp_15 = uilamp(app.LeftPanel);
app.Lamp_15.Position = [416 316 25 25];

% Create Breaker11Switch_2Label
app.Breaker11Switch_2Label = uilabel(app.LeftPanel);
app.Breaker11Switch_2Label.HorizontalAlignment = 'center';
app.Breaker11Switch_2Label.Position = [19 244 63 22];
app.Breaker11Switch_2Label.Text = 'Breaker 11';

% Create Breaker11Switch_2
app.Breaker11Switch_2 = uiswitch(app.LeftPanel, 'slider');
app.Breaker11Switch_2.Position = [31 281 37 16];
app.Breaker11Switch_2.Value = 'On';

% Create Breaker12Switch_2Label
app.Breaker12Switch_2Label = uilabel(app.LeftPanel);
app.Breaker12Switch_2Label.HorizontalAlignment = 'center';
app.Breaker12Switch_2Label.Position = [111 244 64 22];
app.Breaker12Switch_2Label.Text = 'Breaker 12';

% Create Breaker12Switch_2
app.Breaker12Switch_2 = uiswitch(app.LeftPanel, 'slider');
app.Breaker12Switch_2.Position = [124 281 37 16];
app.Breaker12Switch_2.Value = 'On';

% Create Breaker13SwitchLabel
app.Breaker13SwitchLabel = uilabel(app.LeftPanel);
app.Breaker13SwitchLabel.HorizontalAlignment = 'center';
app.Breaker13SwitchLabel.Position = [210 244 64 22];
app.Breaker13SwitchLabel.Text = 'Breaker 13';

% Create Breaker13Switch
app.Breaker13Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker13Switch.Position = [222 281 37 16];
app.Breaker13Switch.Value = 'On';

% Create Breaker14SwitchLabel
```

```
app.Breaker14SwitchLabel = uilabel(app.LeftPanel);
app.Breaker14SwitchLabel.HorizontalAlignment = 'center';
app.Breaker14SwitchLabel.Position = [299 245 64 22];
app.Breaker14SwitchLabel.Text = 'Breaker 14';

% Create Breaker14Switch
app.Breaker14Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker14Switch.ValueChangedFcn = createCallbackFcn(app,
@ResetBreaker, true);
app.Breaker14Switch.Position = [312 282 37 16];
app.Breaker14Switch.Value = 'On';

% Create Breaker15SwitchLabel
app.Breaker15SwitchLabel = uilabel(app.LeftPanel);
app.Breaker15SwitchLabel.HorizontalAlignment = 'center';
app.Breaker15SwitchLabel.Position = [398 245 64 22];
app.Breaker15SwitchLabel.Text = 'Breaker 15';

% Create Breaker15Switch
app.Breaker15Switch = uiswitch(app.LeftPanel, 'slider');
app.Breaker15Switch.Position = [411 282 37 16];
app.Breaker15Switch.Value = 'On';

% Create Disconnect
app.Disconnect = uibutton(app.LeftPanel, 'state');
app.Disconnect.ValueChangedFcn = createCallbackFcn(app,
@Disconnection, true);
app.Disconnect.Text = 'DISCONNECT';
app.Disconnect.BackgroundColor = [1 0 0];
app.Disconnect.Position = [21 474 100 22];

% Create Connect
app.Connect = uibutton(app.LeftPanel, 'state');
app.Connect.ValueChangedFcn = createCallbackFcn(app, @Connection,
true);
app.Connect.Text = 'CONNECT';
app.Connect.BackgroundColor = [1 0 0];
app.Connect.Position = [18 401 100 22];

% Create DatePickerLabel
app.DatePickerLabel = uilabel(app.LeftPanel);
app.DatePickerLabel.HorizontalAlignment = 'right';
app.DatePickerLabel.Position = [238 564 67 22];
app.DatePickerLabel.Text = 'Date Picker';

% Create DatePicker
app.DatePicker = uidatepicker(app.LeftPanel);
app.DatePicker.Position = [320 563 150 23];
app.DatePicker.Value = datetime([2022 7 8]);
```

```
% Create TabGroup
app.TabGroup = uitabgroup(app.LeftPanel);
app.TabGroup.AutoResizeChildren = 'off';
app.TabGroup.Position = [145 355 324 192];

% Create Tab
app.Tab = uitab(app.TabGroup);
app.Tab.AutoResizeChildren = 'off';
app.Tab.Title = '1';

% Create UIAxes_16
app.UIAxes_16 = uiaxes(app.Tab);
title(app.UIAxes_16, 'LED value')
xlabel(app.UIAxes_16, 'Time (tenth of a second)')
ylabel(app.UIAxes_16, 'Digital value')
zlabel(app.UIAxes_16, 'Z')
app.UIAxes_16.YLim = [0 1.5];
app.UIAxes_16.Position = [2 5 316 162];

% Create Tab_2
app.Tab_2 = uitab(app.TabGroup);
app.Tab_2.AutoResizeChildren = 'off';
app.Tab_2.Title = '2';

% Create UIAxes_2
app.UIAxes_2 = uiaxes(app.Tab_2);
title(app.UIAxes_2, 'LED value')
xlabel(app.UIAxes_2, 'Time (tenth of a second)')
ylabel(app.UIAxes_2, 'Digital value')
zlabel(app.UIAxes_2, 'Z')
app.UIAxes_2.YLim = [0 1.5];
app.UIAxes_2.Position = [1 1 316 162];

% Create Tab_3
app.Tab_3 = uitab(app.TabGroup);
app.Tab_3.Title = '3';

% Create UIAxes_17
app.UIAxes_17 = uiaxes(app.Tab_3);
title(app.UIAxes_17, 'LED value')
xlabel(app.UIAxes_17, 'Time (tenth of a second)')
ylabel(app.UIAxes_17, 'Digital value')
zlabel(app.UIAxes_17, 'Z')
app.UIAxes_17.YLim = [0 1.5];
app.UIAxes_17.Position = [1 1 316 162];

% Create Tab_4
app.Tab_4 = uitab(app.TabGroup);
```

```
app.Tab_4.Title = '4';

% Create UIAxes_18
app.UIAxes_18 = uiaxes(app.Tab_4);
title(app.UIAxes_18, 'LED value')
xlabel(app.UIAxes_18, 'Time (tenth of a second)')
ylabel(app.UIAxes_18, 'Digital value')
zlabel(app.UIAxes_18, 'Z')
app.UIAxes_18.YLim = [0 1.5];
app.UIAxes_18.Position = [1 1 316 162];

% Create Tab_5
app.Tab_5 = uitab(app.TabGroup);
app.Tab_5.Title = '5';

% Create UIAxes_19
app.UIAxes_19 = uiaxes(app.Tab_5);
title(app.UIAxes_19, 'LED value')
xlabel(app.UIAxes_19, 'Time (tenth of a second)')
ylabel(app.UIAxes_19, 'Digital value')
zlabel(app.UIAxes_19, 'Z')
app.UIAxes_19.YLim = [0 1.5];
app.UIAxes_19.Position = [1 1 316 162];

% Create Tab_6
app.Tab_6 = uitab(app.TabGroup);
app.Tab_6.Title = '6';

% Create UIAxes_20
app.UIAxes_20 = uiaxes(app.Tab_6);
title(app.UIAxes_20, 'LED value')
xlabel(app.UIAxes_20, 'Time (tenth of a second)')
ylabel(app.UIAxes_20, 'Digital value')
zlabel(app.UIAxes_20, 'Z')
app.UIAxes_20.YLim = [0 1.5];
app.UIAxes_20.Position = [1 1 316 162];

% Create Tab_7
app.Tab_7 = uitab(app.TabGroup);
app.Tab_7.Title = '7';

% Create UIAxes_21
app.UIAxes_21 = uiaxes(app.Tab_7);
title(app.UIAxes_21, 'LED value')
xlabel(app.UIAxes_21, 'Time (tenth of a second)')
ylabel(app.UIAxes_21, 'Digital value')
zlabel(app.UIAxes_21, 'Z')
app.UIAxes_21.YLim = [0 1.5];
app.UIAxes_21.Position = [1 1 316 162];
```

```
% Create Tab_8
app.Tab_8 = uitab(app.TabGroup);
app.Tab_8.Title = '8';

% Create UIAxes_22
app.UIAxes_22 = uiaxes(app.Tab_8);
title(app.UIAxes_22, 'LED value')
xlabel(app.UIAxes_22, 'Time (tenth of a second)')
ylabel(app.UIAxes_22, 'Digital value')
zlabel(app.UIAxes_22, 'Z')
app.UIAxes_22.YLim = [0 1.5];
app.UIAxes_22.Position = [1 1 316 162];

% Create Tab_9
app.Tab_9 = uitab(app.TabGroup);
app.Tab_9.Title = '9';

% Create UIAxes_23
app.UIAxes_23 = uiaxes(app.Tab_9);
title(app.UIAxes_23, 'LED value')
xlabel(app.UIAxes_23, 'Time (tenth of a second)')
ylabel(app.UIAxes_23, 'Digital value')
zlabel(app.UIAxes_23, 'Z')
app.UIAxes_23.YLim = [0 1.5];
app.UIAxes_23.Position = [1 1 316 162];

% Create Tab_10
app.Tab_10 = uitab(app.TabGroup);
app.Tab_10.Title = '10';

% Create UIAxes_24
app.UIAxes_24 = uiaxes(app.Tab_10);
title(app.UIAxes_24, 'LED value')
xlabel(app.UIAxes_24, 'Time (tenth of a second)')
ylabel(app.UIAxes_24, 'Digital value')
zlabel(app.UIAxes_24, 'Z')
app.UIAxes_24.YLim = [0 1.5];
app.UIAxes_24.Position = [1 1 316 162];

% Create Tab_11
app.Tab_11 = uitab(app.TabGroup);
app.Tab_11.Title = '11';

% Create UIAxes_25
app.UIAxes_25 = uiaxes(app.Tab_11);
title(app.UIAxes_25, 'LED value')
xlabel(app.UIAxes_25, 'Time (tenth of a second)')
ylabel(app.UIAxes_25, 'Digital value')
```

```
zlabel(app.UIAxes_25, 'Z')
app.UIAxes_25.YLim = [0 1.5];
app.UIAxes_25.Position = [1 1 316 162];

% Create Tab_12
app.Tab_12 = uitab(app.TabGroup);
app.Tab_12.Title = '12';

% Create UIAxes_26
app.UIAxes_26 = uiaxes(app.Tab_12);
title(app.UIAxes_26, 'LED value')
xlabel(app.UIAxes_26, 'Time (tenth of a second)')
ylabel(app.UIAxes_26, 'Digital value')
zlabel(app.UIAxes_26, 'Z')
app.UIAxes_26.YLim = [0 1.5];
app.UIAxes_26.Position = [1 1 316 162];

% Create Tab_13
app.Tab_13 = uitab(app.TabGroup);
app.Tab_13.Title = '13';

% Create UIAxes_27
app.UIAxes_27 = uiaxes(app.Tab_13);
title(app.UIAxes_27, 'LED value')
xlabel(app.UIAxes_27, 'Time (tenth of a second)')
ylabel(app.UIAxes_27, 'Digital value')
zlabel(app.UIAxes_27, 'Z')
app.UIAxes_27.YLim = [0 1.5];
app.UIAxes_27.Position = [1 1 316 162];

% Create Tab_14
app.Tab_14 = uitab(app.TabGroup);
app.Tab_14.Title = '14';

% Create graph
app.graph = uiaxes(app.Tab_14);
title(app.graph, 'LED value')
xlabel(app.graph, 'Time (tenth of a second)')
ylabel(app.graph, 'Digital value')
zlabel(app.graph, 'Z')
app.graph.YLim = [0 1.5];
app.graph.Position = [1 1 316 162];

% Create Tab_15
app.Tab_15 = uitab(app.TabGroup);
app.Tab_15.Title = '15';

% Create UIAxes_29
app.UIAxes_29 = uiaxes(app.Tab_15);
```

```
title(app.UIAxes_29, 'LED value')
xlabel(app.UIAxes_29, 'Time (tenth of a second)')
ylabel(app.UIAxes_29, 'Digital value')
zlabel(app.UIAxes_29, 'Z')
app.UIAxes_29.YLim = [0 1.5];
app.UIAxes_29.Position = [1 1 316 162];

% Create RightPanel
app.RightPanel = uipanel(app.GridLayout);
app.RightPanel.Layout.Row = 1;
app.RightPanel.Layout.Column = 2;

% Create Image
app.Image = uiimage(app.RightPanel);
app.Image.BackgroundColor = [1 1 1];
app.Image.HorizontalAlignment = 'left';
app.Image.Position = [158 220 548 176];
app.Image.ImageSource = fullfile(pathToMLAPP, 'GenerationUSA.PNG');

% Create Image2
app.Image2 = uiimage(app.RightPanel);
app.Image2.Position = [157 38 550 178];
app.Image2.ImageSource = fullfile(pathToMLAPP,
'GenerationPyron.PNG');

% Create Image4
app.Image4 = uiimage(app.RightPanel);
app.Image4.Position = [158 401 548 181];
app.Image4.ImageSource = fullfile(pathToMLAPP,
'GenerationWorld.PNG');

% Create ScaleKnobLabel
app.ScaleKnobLabel = uilabel(app.RightPanel);
app.ScaleKnobLabel.HorizontalAlignment = 'center';
app.ScaleKnobLabel.Position = [61 437 35 22];
app.ScaleKnobLabel.Text = 'Scale';

% Create ScaleKnob
app.ScaleKnob = uiknob(app.RightPanel, 'discrete');
app.ScaleKnob.Items = {'Pyron', 'Texas', 'USA', 'World'};
app.ScaleKnob.Position = [54 474 49 49];
app.ScaleKnob.Value = 'World';

% Create ScaleKnob_2Label
app.ScaleKnob_2Label = uilabel(app.RightPanel);
app.ScaleKnob_2Label.HorizontalAlignment = 'center';
app.ScaleKnob_2Label.Position = [61 258 35 22];
app.ScaleKnob_2Label.Text = 'Scale';
```

```
% Create ScaleKnob_2
app.ScaleKnob_2 = uiknob(app.RightPanel, 'discrete');
app.ScaleKnob_2.Items = {'Pyron', 'Texas', 'USA', 'World'};
app.ScaleKnob_2.Position = [54 295 49 49];
app.ScaleKnob_2.Value = 'USA';

% Create ScaleKnob_3Label
app.ScaleKnob_3Label = uilabel(app.RightPanel);
app.ScaleKnob_3Label.HorizontalAlignment = 'center';
app.ScaleKnob_3Label.Position = [61 75 35 22];
app.ScaleKnob_3Label.Text = 'Scale';

% Create ScaleKnob_3
app.ScaleKnob_3 = uiknob(app.RightPanel, 'discrete');
app.ScaleKnob_3.Items = {'Pyron', 'Texas', 'USA', 'World'};
app.ScaleKnob_3.Position = [54 112 49 49];
app.ScaleKnob_3.Value = 'Pyron';

% Create EditField
app.EditField = uieditfield(app.RightPanel, 'text');
app.EditField.HorizontalAlignment = 'center';
app.EditField.BackgroundColor = [0.9412 0.9412 0.9412];
app.EditField.Position = [605 9 100 23];
app.EditField.Value = '© RWE 2022';

% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end

% App creation and deletion
methods (Access = public)

% Construct app
function app = finalapp

% Create UIFigure and components
createComponents(app)

% Register the app with App Designer
registerApp(app, app.UIFigure)

% Execute the startup function
runStartupFcn(app, @startupFcn)

if nargout == 0
clear app
end
end
```

```
% Code that executes before app deletion
function delete(app)

% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end
end
```