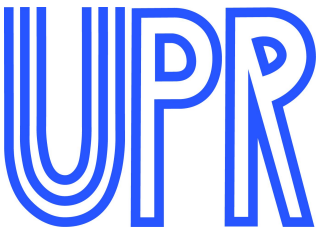
Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual
Utility Patrol Robot

Submitted to:

Sergo Sagareli, PE, PMP
Secretary, CIGRE WG B3.47
Con Edison of NY, Inc.
4 Irving Place, Room 2215-S
New York, NY 10003
212 460-2284 (O) 646-662-076 (M)

By



Team 24
UPR Level Security

Team Members

Eugenia Almandoz eugeniaa@bu.edu
Namir Fawaz namirfa@bu.edu
Brock Guzman brockg@bu.edu
Juan Peralta jperalta@bu.edu
Brandon Webster bweb@bu.edu

Submitted: 4/30/18

Utility Patrol Robot

Table of Contents

Executive Summary **2**

1 Introduction **3**

2 System Overview and Installation **5**

2.1 Overview block diagram 5

2.2 User interface. 5

2.3 Physical description. 5

2.4 Installation, setup, and support 5

3 Operation of the Project **6**

3.1 Operating Mode 1: Normal Operation 6

3.2 Operating Mode 2: Abnormal Operations 6

3.3 Safety Issues 6

4 Technical Background **7**

5 Cost Breakdown **8**

6 Appendices **9**

6.1 Appendix A - Specifications 9

6.2 Appendix B – Team Information 9

Executive Summary

UPR Level Security was formed to design a utility patrol robot to more effectively monitor substations. Electric companies, such as Con Edison, are looking for a product like this to detect unauthorized personnel in the station and detect anomalies in the equipment that could potentially have disastrous results if left unchecked. One way to achieve this goal is by placing a utility patrol robot in the substation to monitor it daily. A utility patrol robot is preferable to deploying human security personnel because it can operate more often and is a safer solution than sending a human into the substation.

In order to provide Con Edison with a solution to this problem, UPR Level Security will be providing them with one fully functional patrol robot and the software needed to operate it, to be used as a prototype. The software includes a web application to interact with the robot, algorithms necessary for autonomous navigation, and software for image recognition. To solve the problem at hand we will be using a combination of object detection and localization to autonomously navigate the robot around the substation. The robot will be mounted with ultrasonic sensors, a GPS, and a camera. The ultrasonic sensor will be used to detect objects, and the camera will be used to read gauges, stream live video, and detect a person. Using a Raspberry Pi connected to WiFi we will be storing information with a server hosted using AWS and communicating that to the user through a web application to alert the user of any anomalies the robot detects. Through the web application a user will also be able to remotely control the robot manually to do specific tasks. When comparing our product to similar electrical substation robots found on the market, an innovative feature our product provides is security. In addition to detecting faults in the equipment our robot looks for unauthorized personnel in the station.

1 Introduction

ConEdison's Sergo Sagareli proposed this Utility Patrol Robot project to our team to provide a system that would make regular maintenance and fault checks throughout their electrical substations while decreasing the risk of employees working near high voltage equipment.

Electrical substations provide alterations to the voltage between transmission and distribution sections of the power grid using machinery that handles high levels of voltage. These substations step voltage up from distribution to transmission, and vice versa, using transformers to ramp up the voltage for transmission, thus decreasing the current load that needs to be transported, or decreasing the voltage to safer distribution levels around residential areas. This means that stations, such as ConEdison's, use machines rated at 345 kV which can carry dangerous levels of current, can experience faults that could be life threatening, such as arcing where electricity is released into the air at temperatures up to 2500 degrees Celsius, and faults that could damage the very expensive equipment. These stations undergo regular maintenance checks to ensure their functionality and security, yet it's still difficult to predict and then prevent faults from occurring. Electrical substations are large and tend to be placed outside, so they are subject to the weather and geography with rough surfaces and various degrees on inclination.

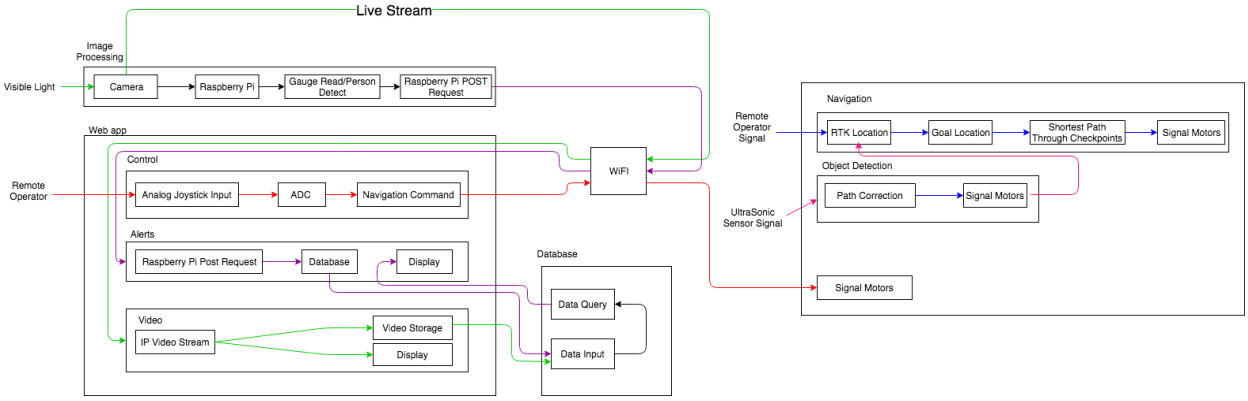
Our team's project will solve several problems in the safety of regular maintenance checks while also providing a method of early fault detection. Our patrol robot will provide similar gauge measurements as employees without putting people in potentially dangerous situations, and will frequently survey the station for faults and security breaches. Our team's approach to provide solutions to these problems is a patrol robot that can navigate autonomously throughout a substation and its varying terrain, read the gauges on the machines, stream live video to a web server, detect unauthorized people, provide remote override of the robot including control, and report all detected alerts to the server.

The automated navigation of the robot will establish the necessary mobility to travel throughout the substation, which is the basis of our robot's functionality. The gauge readings will provide multiple measurements a day to the control station to detect for irregularities and ensure proper operation, without the need to deploy an employee. Live stream video will serve as a security recording in case of breaches or faults, and will also serve as the eyes to a remote operator using the navigation override controls. In the case that the control station operator cares to observe sections of the substation, he or she can control the robot throughout these areas without needing to be present in the station. Detection of humans will alert the control station of potential sources of

equipment damage, saving the company a potential loss of machinery. Lastly, the robot's various sensors will detect faults, like partial discharges, before they grow into larger issues. The detection of such situations will be sent to the control station so they may take appropriate actions before to prevent larger problems.

2 System Overview and Installation

2.1 Overview block diagram



The project can be broken down into a few major components which are described in the block diagram above. The first component is the Image Processing, which allows the robot to detect and read gauges and send the results to our AWS-hosted Web Application.

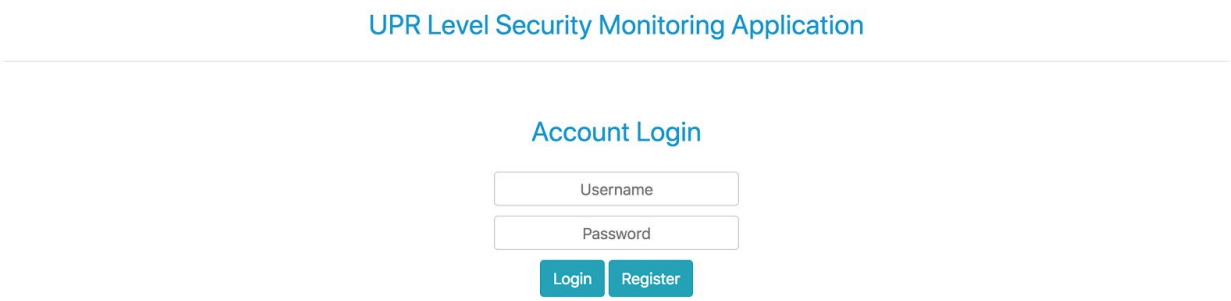
The next major component is the Web Application/Database, which is used for multiple purposes. The first reason is to display all of the data in our database, from gauge detections and readings to unauthorized people alerts. The next use is to view the livestream of the robot that the Raspberry Pi sends to the Web App. Finally, the Web App is also used to remotely control the robot whenever necessary.

Finally, the last component of our project is the navigation itself. The robot uses its GPS location in tandem with the map of locations that have been uploaded to the Raspberry Pi, and navigates in a straight line towards the destinations. In case there is some obstruction in the way, the robot uses Ultrasonic Sensors to do object avoidance, before recalculating a new path towards its destination. The motor of the robot as well as the object avoidance system are both controlled by the Arduino.

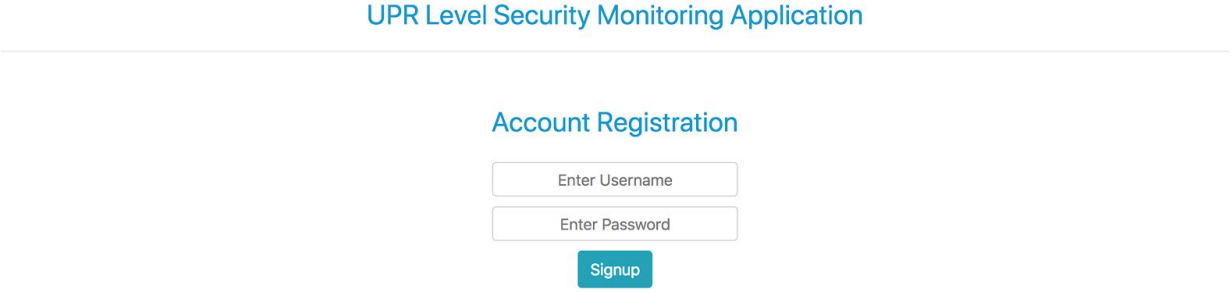
2.2 User interface.

The majority of human interaction with our robot comes through the Web Application. The user interface is very simple and easy to use.

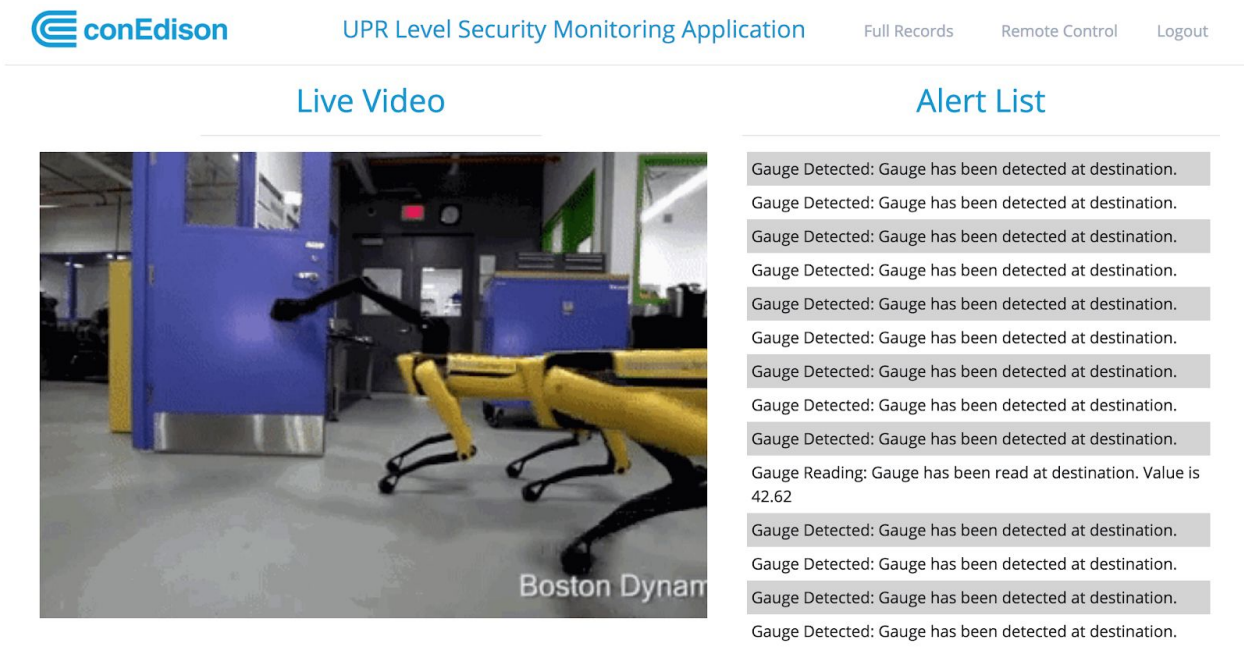
The first page is the Login page, which is very intuitive and simple:



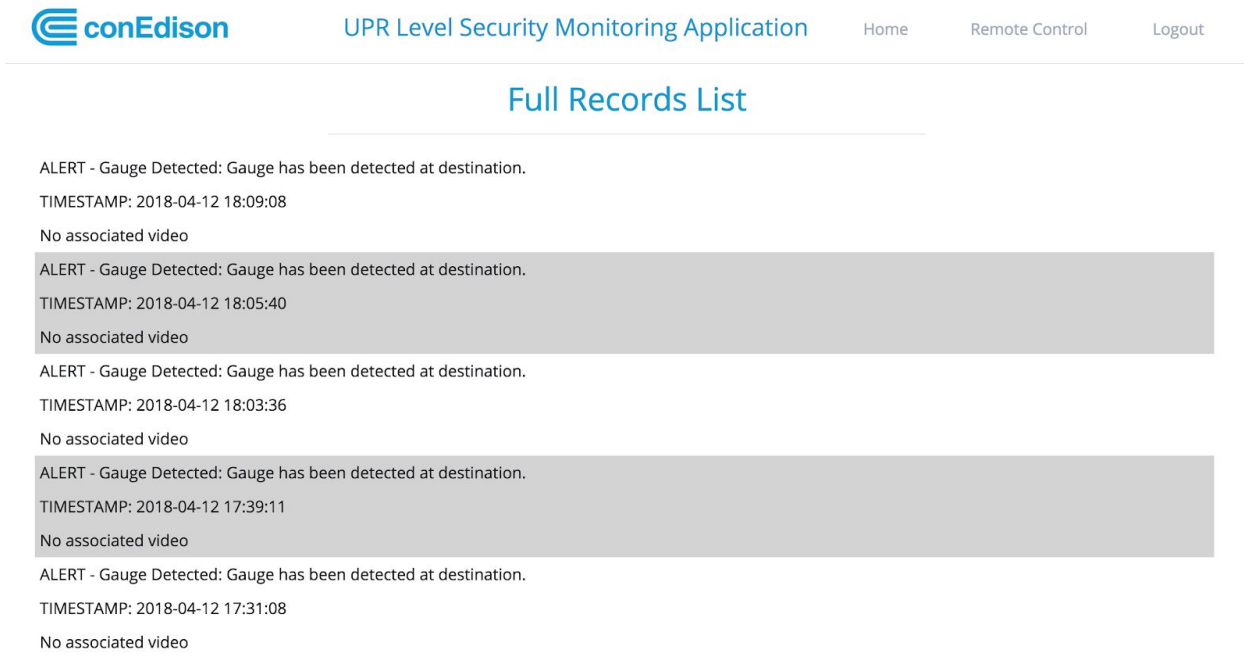
The next page is the Registration page. This of course will not be included in the final product, but is simply a proof of concept showing the functionality of creating a new user.



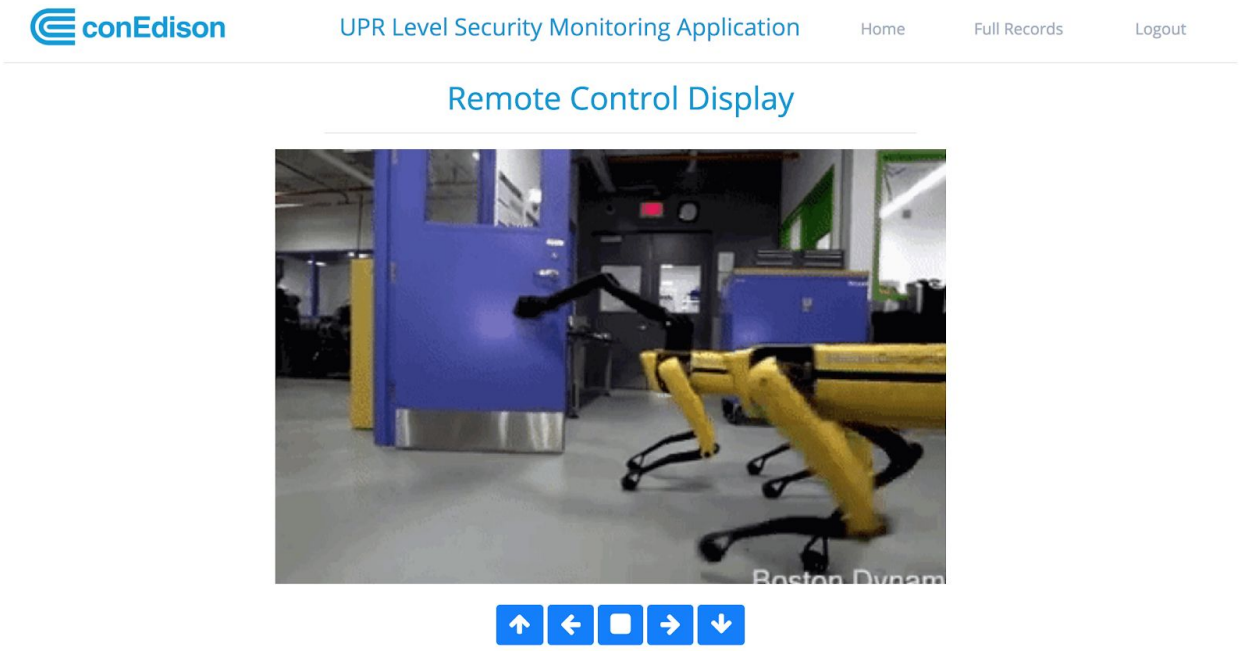
Next there is the home page, which displays the most recent Alerts, as well as a live view of the robot.



There is also the Full Alerts List, which displays all of the alerts that have ever been recorded by the robot, along with an associated video, a timestamp, and an alert title/description.

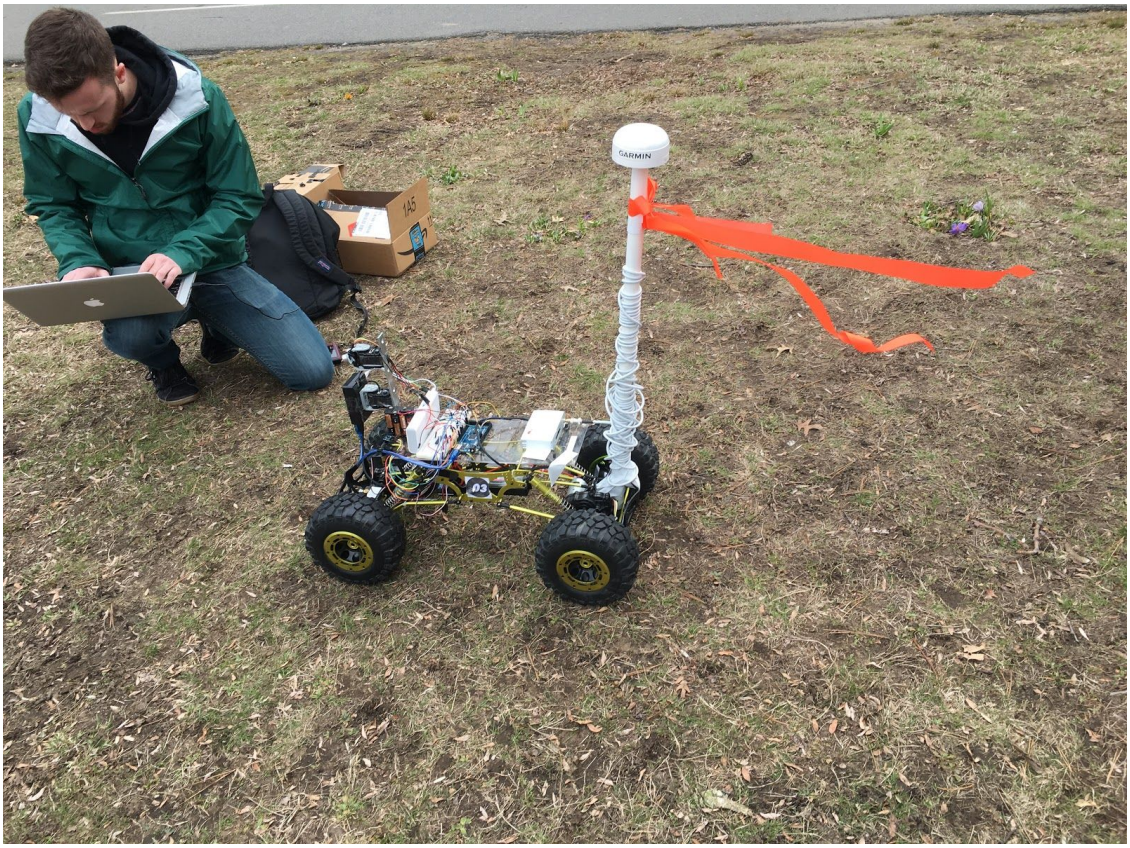


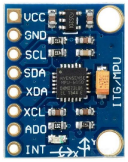
Finally, there is the Remote Control page, which allows the operator to control the robot using the WASD keys. This feature disables all autonomous capabilities of the Robot.



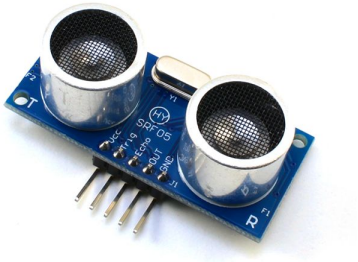
2.3 Physical description.

Full Project

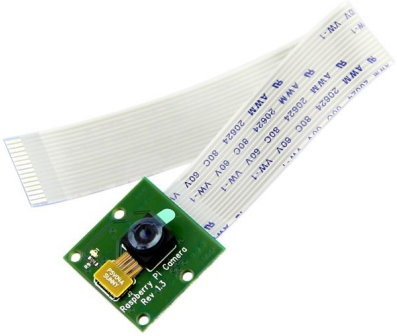




MPU6050



HY-SRF05 Ultrasonic Sensor



Raspberry Pi 3 Camera Module

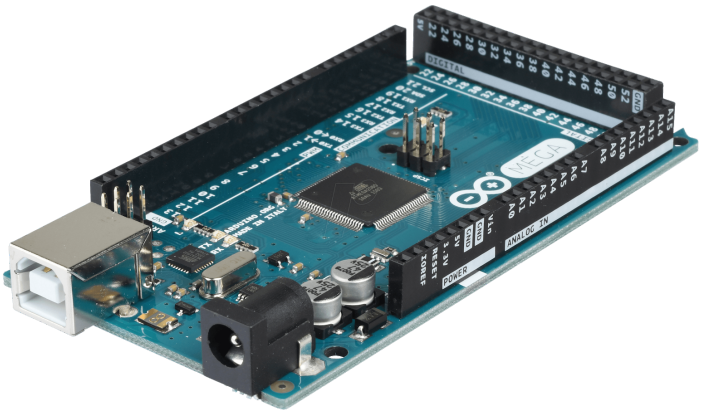
Sn74ls241n Tri-state Buffer

Raspberry Pi 3 B

Dynamixel AX-12A Smart Servo Motor

U-Blox M8 GNSS

Garmin Antenna



Arduino Mega

All components together on the Exceed RC Rock Crawler Radio Truck

2.4 Installation, setup, and support

To run the web application on Amazon Web Services, an AWS account with the appropriate IAM security permissions is required. Once the IAM permissions are created, access keys will need to be generated on the Raspberry Pi, in the PersonAlert.py file. For the web application, the Elastic Beanstalk application (created using the AWS console and deployed to with either the AWS CLI or EB CLI) will need to be explicitly given permission to access S3 so that it can serve the patrol videos on the server. A Bucket will need to be created in S3 to host the video files created from patrols. There is a python script included with the web application code that will automatically create this Bucket when run.

The Raspberry Pi is set to run it's operational code and server upon startup, using crontab and a bash script (the "run_server" script located on the Desktop). Any output for the code useful for debugging is logged into /tmp/serverinfo.txt. Due to the multithreaded nature of this code, the log file may be written out of order. Logs are overwritten after every system reboot.To connect the Raspberry Pi to a new wifi network, a monitor, mouse, and keyboard must be used. Once connected to a network, the Raspberry Pi can also be accessed through SSH, which is the preferred method of access once the Raspberry Pi is placed upon the robot. Within the Raspberry Pi code, the IP address (which can be acquired with the command line call "hostname -I") will need to be changed at the bottom of the main.py file to the current IP address.

If installing the Raspberry Pi code on a fresh Raspberry Pi, a virtual environment and associated requirements will need to be installed so that the server can run successfully. These requirements may be viewed and installed from the requirements.txt file. Additionally, both Python 2 and Python 3 must be available on the device. Python 3 should be the default installation.

The Arduino currently contains its operating code in its memory. If this code needs to be changed, upload it using the Arduino application from a laptop or desktop computer (not the Raspberry Pi, which is unreliable in uploading this code). The uploading computer can also be used to debug the Arduino through the serial connection.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

There are two modes of operation of the robot: Autonomous Mode and Remote Control. It also has a kill switch option to freeze all motor functionalities. When the raspberry pi script is started it will automatically be in autonomous mode. This means it will navigate to the pre-installed GPS coordinates. On the web application you can click on the remote control tab to switch to remote control mode, which may take a couple of seconds to establish but the robot should stop autonomous mode as soon as it is clicked. To move the robot you will need to use the WASD keys and the ZC keys on your keyboard:

W - Forward

A - Forward Left

D - Forward Right

S - Backward

Z - Back Left

C - Back Right

ESC - Brings you back to autonomous mode

While currently there is not kill switch button on the web application, the robot is configured to receive a kill switch parameter at any moment just like remote control, which takes priority. This feature would be implemented the same way remote control was implemented.

3.2 Operating Mode 2: Abnormal Operations

Abnormal operations of the patrol robot occur when the GNSS is not fully accurate but consistent, ie. it may not return the exact location of the device but is consistently within a meter or two of the location it receives. When this occurs the robot uses differential GNSS to correct the positioning by considering known distances between locations. At startup the code checks if the received lat/lon is consistent with a value it knows and if not it will take the difference and modify the goal locations with the same distance. This will at least place the robot in an area where it can scan for gauges with its camera and move closer on its own.

If autonomous navigation is not available for reasons not yet known, there is a human operation mode that allows for a controller to take over the navigation via the web page. This will allow for safe avoidance of any obstacles such as expensive or dangerous equipment in the case where the robot’s navigation is faulty. An operator will be able to see live video stream from the robot’s camera and can control its navigation with key strokes on his or her computer.

Another abnormality in operations comes from a poor calibration of the MPU6050 - an accelerometer/gyroscope used for relative positioning. If the calibration of this unit goes poorly,

the robot will read its direction as incorrect because the MPU's readings will keep changing. To deal with this a timer measures the duration between startup and the "Calibrated" message from the Arduino. If this duration is either too long or too short the Raspberry Pi will reset the Arduino, thus restarting the MPU6060 calibration. While this does not occur often anymore, this failsafe ensures that the robot receives accurate readings throughout its activity.

3.3 Safety Issues

All safety issues from our project arise from catastrophic system failures. The purpose of our autonomous robot is to reduce the need for humans to maintain Con Edison's electrical substations, because of the inherent risk that comes from being near such high voltage equipment. As a result, if the robot were to malfunction in a fundamental way, for example if it were to crash into a component of the electrical substation, it could cause a lot of damage. The results of this could be very serious, potentially causing fires or creating power outages in the New York City area. Our team has worked tirelessly to make sure that such a malfunction would be impossible, and that in all cases the robot would simply stop moving and shut down if it behaves incorrectly.

Another potential risk could involve the security of our Web Application. There is no data on the Web Application that would be considered sensitive information, and so there is effectively no risk that someone could hack into the site in order to steal data. The one major risk, however, is that someone could hack into the site and begin to remotely control our robot. This would be very dangerous, as the hacker could cause an accident in the electrical substation. Our team is confident that this will not be possible, and are working to have the Web Application hosted on a more secure server in order to address this concern.

4 Technical Background

Navigation:

The final design for the navigation of the design includes a GNSS-based, shortest distance navigation module, an Ultrasonic-Sensor-based object avoidance module, a camera-based gauge finding module, and a remote control module in the event that an operator wants to control the navigation.

The GNSS-based navigation takes measurements from a U-Blox m8 GNSS Evaluation Kit connected to a Garmin antenna* and compares it to a known starting location of the robot (assuming it starts in the same spot each time), then corrects it with known distances, such as its known location subtracted from its measured location, the known location of MassDOT towers and the location received from the robot's ntrip connection to the MassDOT site. The robot makes the same corrections with known locations of gauges throughout the station. With these new values the robot makes a calculation to find the shortest path to the gauges, using its initial orientation (hard coded based on site specifications and translated to work with the MPU6050) to calculate how far in which direction the robot needs to turn to travel in the shortest path to the goal location. The robot is the signaled to turn slowly until the MPU6050 measures that it has turned a proper amount and the robot will proceed straight. If for any reason the MPU6050 measures anything different than the proper angle, the robot will receive signal to correct itself.

If at any time the Ultrasonic sensor measures that something is obstructing its path (currently set at .5 meters), it will begin its object avoidance module to continue onto its shortest path. The robot will receive signal to stop its motors and scan its path in 15 degree increments with its smart servo, taking ultrasonic measurements at each position. It then compares the measurements to a "clear path threshold" (1.5 meters currently) and then finds the position that is closest to the current path. It will then signal to it steering servo to turn in that direction, travel for a short amount of time to avoid the object, then recalculates its shortest path and travels in that direction. If that path is blocked it will continue the process until it is free. As of now we are assuming that the robot will not find itself in a corner that is impossible to get out of without retracing its path, this assumption is based on given substation layout.

Once the robot is within a 1.5 meter box of the goal location, it will move into its gauge finding module. This begins by stopping the robot's motors then finding searching for the gauge with the camera and smart servo. Once the gauge detection software on the Pi finds the gauge, the smart servo will slowly center the gauge in its frame, then relay that angle to the MPU and the steering servo to begin the slow navigation to the gauge. Once the gauge angle is read by the Pi, the robot will await the next location to which it will travel.

The final navigation module is the remote control operation. In this section of navigation, a user on the web page will be able to use his or her keyboard to control navigation of the robot. The Web Application uses Javascript functions to record when a button has been pressed down,

and when the button has been released. As long as the button is pressed down, the Web App will continue to send the signal corresponding to that button. When the button is released, it immediately sends a stop signal to the Raspberry Pi, signalling that the button is no longer pressed and the robot should stop moving. Based on the key command, the robot will either send a left, right or straight command to its steering servo, and a forward, backward or stop command to its motors. The keys used for remote control are “w”, “a”, “s”, “d”, “z”, “c”, and escape. The escape key is used to send the robot a command that will end remote control and return to autonomous navigation. The other keys are used to send the robot the following commands:

Key	Action
W	move forwards
A	move forward and turn left
D	move forward and turn right
S	move straight backwards
Z	move backwards and turn left
C	move backwards and turn right

When the key is pressed down, a begin command is sent to the robot using an Ajax request pointed at the server running on the Raspberry Pi. When the key is released, a stop command is sent using the same method.

Imaging:

The final product includes two main imaging components: person detection and gauge detection. All the imaging was done on the Raspberry Pi 3 B using the OpenCV libraries for python and using the Pi camera module.

Person Detection:

The person detection was achieved by using a built in opencv person detector which is a pre-trained SVM(support vector machine) classifier based on HOG(histogram oriented gradients) features. The person detection is set to be on during autonomous navigation. This excludes the part where the robot has reached its target and is only performing gauge detection.

Once a person is detected it sends an alert to the web application, which will be explained in the web application segment. Also once the person is detected it feeds the region of interest, rectangle coordinates, to a KCF(Kernelized Correlation Filters) tracker.

KFC Tracker:

The tracker is used because it reduces the CPU usage as it is less computational extensive as a detector, and it is able to continue to track the detected person even when the detector fails. The tracker checks for the object every 15 frames to make sure it is not tracking an incorrect object. The tracker also is able to send commands to the arduino to move the camera servo in order to center the target on its frame. This part is done for the gauge detection part discussed below, and could be implemented for person detection or other object detection modes that are applied.

Gauge Detector:

Once the robot reaches the GPS destination gauge detection is activated. Since the gauges that we would be looking for would be circular, and other circular object would not be near the desired gauge, we used the Hough circle transform for gauge detection. This is a built in opencv function and by adjusting the parameters we were able to get a reliable detection distance of 2 meters of a 10 inch clock, which we used since it had similar characteristics as the desired gauge. An actual gauge would be smaller than that size which means the reliable distance would fall. The GNSS system we used for localization was able to achieve distance of around 2 meters which worked for demo purposes, but on a substation environment we would have needed to be able to successfully implemented the GNSS with RTK corrections, which would have provided accuracies up to 1cm.

Gauge Reading:

To be able to read the value of the gauge, reading the angle of the needle would need to be done. We accomplished this by using opencv function HoughLinesP(). Once the gauge was detected, we perform the transform on the ROI. Using the coordinates of the line output given by the function we are able to know the angle of the line. The line detected in the needle of the clock we used. We were able to get reliable readings within 30 cm. From knowing the angle of the clock you would then need to have a function that transcribes these angles to actual values. For a clock you know each hour is 30 degrees from each other. Given more time we would have easily implemented this aspect so that the alert that is seen on the web application also gives the actual reading value of the “gauge”. For differentiating between quadrants given the angle, you can easily just look at the location of the line coordinates given by the function and know which quadrant you are in, which means you can differentiate between for example 12 o’clock and 6 o’clock.

Web Application:

The Web Application is hosted on AWS using Elastic Beanstalk, RDS, and Amazon S3. It was built using the FLASK Stack. The FLASK Stack was chosen because it is based in Python, and allowed us to use Python for the Web Application as well as the Raspberry Pi. The Web Application is the main hub of information regarding every aspect of the robot. The database was created using SQL Alchemy, and is used to store information including all alerts sent from the robot, all associated video clips, and all of the login credentials for authorized users. The database uses a main API named “alert” which allows the Raspberry Pi to remotely send the title and body text which describes the alert, and the associated video if it exits. The database also has unique model types for each different kind of alert. The Web Application has four main pages, the Home page, the Full Records page, the Remote Control page, and the Login/Registration page, (registration is only included as a proof of concept, in reality users would have to be explicitly created if they are authorized to access the Web App). All of the pages are created using HTML and CSS, and additionally contain some front-end Javascript work if necessary. The Home page and Remote Control page both contain a live stream of the robot, which is displayed using an istream from a secured IP address that the Raspberry Pi pushes the video to. The Remote Control page’s functionality has been explained in an earlier section. The Full Records page displays all of the alerts that have ever been sent to the Web Application, including a timestamp, a title, a body, and a link to download the relevant video clip if available. The Home page shows a shorter list of the most recent alerts.

5 Cost Breakdown

Item	Description	Cost
1	Exceed RC Rock Crawler Radio Truck	$200
2	MPU6050	$6
3	HY-SRF05 Ultrasonic Sensor	$4
4	Raspberry Pi 3 Camera Module	$30
5	Sn74ls241n Tri-state Buffer	$2
6	Raspberry Pi 3 B	$30
7	Dynamixel AX-12A Smart Servo Motor x2	$90
8	U-Blox M8 GNSS	$179
9	Garmin Antenna	$57
10	Arduino Mega	$38
11	Battery	$40
	Total	**$676**

The budget is dominated by the crawler as well as the navigation aspect of it, and rightfully so since it was given as the most important functionality. Our cost should have been much higher but given the time constraint of when we got our budget finalized up until the end of the project we kept things as simple as possible. We used resources we already had access to maximize the amount of time we were able to work on the project. Some of the costs we should have included to completely meet our customers requirements include a thermal camera which was around $250. Trips to visit the substation which averaged out to $1500 were also discarded due to time constraints as well as conflicts with free time with our client. Lastly, if we had more time between the finalizing of our budget and the end of the project we would have looked into better microcontrollers as well as more efficient sensors to make the outcome overall more seamless.

6 Appendices

6.1 Appendix A - Specifications

Give these as a list or table. These should quantify the performance that the project provides, as it is built. Do not simply repeat functional requirements here. These are the final delivered specifications. These will usually map closely to the User's requirements.
1 page, table format

6.2 Appendix B – Team Information

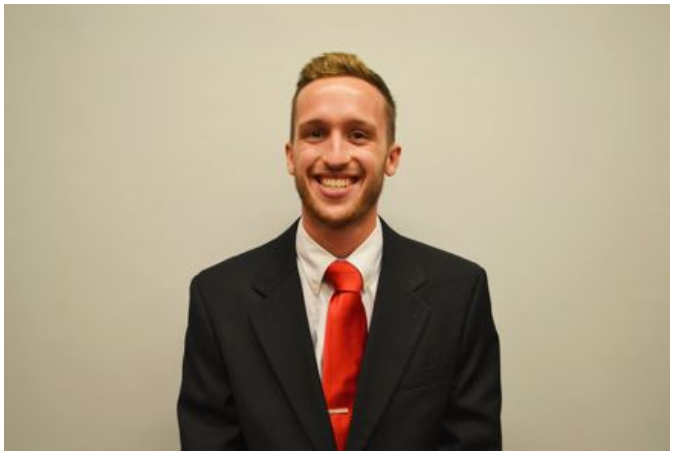
Utility Patrol Robot Team Members

.

Eugenia Almandoz:

Eugenia is majoring in Computer Engineering and has experience with microprocessor programming, networking and software languages such as C++, C, C#, Python and some Java. She will be the Product Manager for the team as well as a developer for the Utility Patrol Robot Project. Eugenia will be joining Cisco after graduation under their Sales Associate Program.

Brandon Webster:



Brandon is majoring in Computer Engineering, and has experience in image recognition and image processing, along with general programming skills such as Object Oriented Programming, with familiarity in C, C#, C++, Java and Python. I also have some experience in coding for robotics, particularly as it relates to pathfinding and navigation. I will be taking a software developer role on our team for this project. Brandon will be working as a Software Engineer for Paytronix Systems after graduation.

Namir Fawaz:

Namir is a Senior at Boston University majoring in Computer Engineering. He has a strong background in programming with languages such as C/C++, Python, C#, and many others. He has a great understanding of CPUs and computer hardware, and has worked with GPUs to develop high performance programs using multithreading. Additionally, he has some experience with machine learning and neural networks. He will be working as a software developer for the Utility Patrol Robot project

Brock Guzman:

Brock Guzman is an Electrical Engineering major getting a minor in Computer Engineering. He has experience with circuitry and soldering, microprocessor programming, and software development with Python, C++, and C. He has significant experience in various tools like Xilinex and PSPICE. He will be working on the Embedded Systems and Robotics.

Juan Peralta:

Juan is majoring in Electrical Engineering. He has a good background with digital signal processing and analog electronics. He will be working as a hardware developer for the project.