Senior Design
ENG EC 463

Memo

UPR

To: Professor Pisano

From: UPR Level Security: Juan Peralta, Eugenia Almandoz, Namir Fawaz, Brock Guzman, Brandon Webster

Team: 24

Date: 11/14/17

Subject: First Deliverable Test Plan

1.0 Object Detection - Arduino

Goal:

In order to get the patrol robot navigating, it will first need to have an object detection mechanism. Using an arduino as the controller, which will also be the controller for the robot, we have set up three ultrasonic sensors which will detect if there is any object within a given distance of the sensor. The sensors are setup to detect an object in front, to the left, and to the right of where they are placed. This is the first basic step for an object to know its surroundings.

Procedure:

The three sensors are setup on a box, which will act as the robot in a stationary state. The sensors are connected to an Arduino via its digital pins, each using four pins for power, ground, trigger, and echo. The Arduino calculates distance of an object based on the time it takes for the triggered sound wave to echo back to the sensor. Objects will be placed around the sensors to check for detection.

Result:

The Arduino's serial monitor should receive the following messages depending on what sensors detects an object:

-Object detected within 10cm of its left sensor
-Object detected within 10cm of its right sensor
-Object detected within 10cm of its front sensor

Multiple sensors can go off at the same time, meaning multiple messages can be displayed on serial monitor simultaneously. The Arduino checks for an object every 1 second. If no object is detected, the monitor does not receive any alerts.

2.0 Communication between Arduino and Raspberry Pi

Goal:

Having the Arduino and Raspberry Pi be able to communicate is crucial for being able to send alerts and receive commands for the robot to follow. In this test we will be getting the Arduino to send information to the Raspberry Pi. The Arduino will be set up as it was in the first test, it will be receiving alerts for when an object is within 10cm of any of the three sensors. The goal is to get the Raspberry Pi to receive these alerts.

Procedure:

As stated above, the Arduino is set up similar to the previous test, except the serial port is connected to the Raspberry Pi via USB instead of a computer. On the Raspberry Pi's end we write code in the Python 2. We first look for what port the Arduino is connected to using shell command : *ls /dev/tty**

Then we ask the Raspberry to import serial information from the port at 9600 baud rate : *import serial*

ser = serial.Serial('/dev/ttyACM0', 9600)

Finally we ask it to read the lines it receives continuously and print them:

```
while true:
    read_serial = ser.readline()
    print read_serial
```

Result:

We expect to receive the same alerts the Arduino's serial monitor received on the last test, but this time they appear on the Raspberry Pi's shell after running the python program which includes the code above. Shell alerts:

-Object detected within 10cm of its left sensor
-Object detected within 10cm of its right sensor
-Object detected within 10cm of its front sensor

It will scan for object every one second, and multiple sensors can go off at the same time. Objects will be placed around sensors to test this functionality.

3.0 Web Application

Goal:

The purpose of our web application was to have a central point for the user to navigate to for basic control of our robot. The application will be the place where the user can get information such as live video, alerts on different faults the robot detects in the substation and finally a way for the user to override the robot and control it manually. For our first deliverable we implemented the live video being streamed from the Raspberry Pi camera, an alert bar that senses basic detections from our sensors and finally the ability to start and stop the video.

Procedure:

In order to implement our website we used a combination of HTML and flask which is a python based micro framework. We also implemented a SQLAlchemy database to be able to store the alerts from the pi and be able to display them to our website. In order to run the website the following commands must be run to install everything necessary and get the website running:

-To be run in application directory

- pip install -r requirements.txt
- export FLASK_APP=application
- export FLASK_DEBUG=true
-pip install -e .

- To run the webapp you type flask run and the website will be running on http://0.0.0.0:5000/

Result:

As a result the website will display a live video on the left side of the screen with a start/ stop button in order to control the video feed and an override button that will eventually be used to take the user to another screen to control the robot manually. On the right side of the screen the alerts bar is shown which gives the user short descriptions of what the robot is detecting. This is the basic functionality of our website and we will be building on this model. Next steps will involve ability to manually control the robot as well as the capability to start or shut off the robot regardless of location.

4.0 Video Feed from Raspberry Pi to Web App

Goal:

One of the security tasks given is to send a live video feed to the remote operator. We can accomplish this by integrating a camera to the Raspberry Pi, and sending that video feed to the web application. The goal of this test is get the camera to stream a live video and then getting the video feed to show on the web application.

Procedure:

First we connect the camera to the Raspberry Pi's camera port. The camera is then configured by the Raspberry Pi, and then using a mjpg streamer it sends the video to an ip address. The web application then uses istream to play the video from the specified ip address.

Result:

The web application will be displaying live video being taken from the camera module on the Raspberry Pi.

5.0 Alert Feed from Raspberry Pi to Server to Web App

Goal:

Considering our robot's need to relay information about the conditions of a substation we decided to implement an alert feed on our website to inform the users of important events they must further investigate. As mentioned above, we used the SQLAlchemy database to store events detected by the sensor via the Raspberry Pi. We are then able to query the database to gather these alerts and display them on our website using a list in HTML.

Procedure:

In order to communicate the readings from the sensors to the web application the arduino is connected serially to the Raspberry Pi (via usb) and the Raspberry Pi reads the serial signal sent from the arduino. Given this alert from the arduino, the Raspberry Pi creates a JSON Post request which is sent to the server at (server_IP_address):5000/api/alert. This post request if received by the API, which in turn decodes the json and creates an entry in the database. The web application reads these and populates an alert list in the website. These alerts have a title and a text portion associated to it, to give the user a more detailed explanation of what the robot is detecting. As stated, in the HTML we take the collection of alerts and display them to the website using an unordered list by iterating through all of the alerts in the database.

Result:

As of now this displays the alerts in a bullet style manner to our alert feed simply stating what the sensors have detected. For a more advanced future implementation we plan to display alerts detected at that moment in the middle screen and want to make each alert accessible to display more detailed information to the user. These alerts will be associated with pictures/ video of the detected fault and will be associated with a time stamp as well to give the user a sense of what has been going on in the substation in their absence.