**DAT CAR (Data Acquisition Through a Car Altering Remote)**

**Brandon Davis & Gage Kettering**

**Final Draft**

**April 21st, 2020**

**CEN 4935 Senior Software Engineering Project**

**Instructors: Dr. Huzefa Kagdi, Dr. Janusz Zalewski**

**Department of Software Engineering**

**Florida Gulf Coast University**

**Ft. Myers, FL 33965**

Copyright © 2019 by Florida Gulf Coast University

|  |  |  |
| --- | --- | --- |
| **Individual Student Contributions to the Project** | | |
| **Student Name** | **Percentage (%) of Contribution to a Particular Phase** | **Remarks (optional)** |
| **SOFTWARE SPECIFICATION** | | |
| Brandon Davis | **49%** | Good performance |
| Gage Kettering | **51%** | Excellent contribution |
| **SOFTWARE DESIGN** | | |
| Brandon Davis | **51%** | Excellent contribution |
| Gage Kettering | **49%** | Good performance |
| **IMPLEMENTATION AND TESTING** | | |
| Brandon Davis | **50.5%** | Immense Contributions |
| Gage Kettering | **49.5%** | Tremendous Contributions |

**1. Introduction**

**1.1 2019 Project**

**1.1.1 Project Overview**

There has been a growing demand for data collection in remote environments. This includes environments not accessible to humans or environments too hot, cold for human interaction, or even environments that have radioactive residue near them. Through a remotely controlled vehicle, one may have the ability to access environments that have yet to be accessed or unable to be accessed by humans in general. This device is designed to gain environmental knowledge on areas a human cannot interact with. The adopted solution named DAT CAR, which is much smaller than a human, offers the ability to evaluate situations humans cannot. The user can control the car from a remote location of their choice using a WiFi connection. For example, if there was a fire that burned down a building, DAT CAR can traverse the tight environment and relay data based on the environment around it, while the user is in a safe environment nearby.

DAT CAR is equipped with a camera that displays what is in front of the car via a live feed. There are also sensors that relay the temperature of the environment as well as the humidity. In order to detect the distance to an object, an ultrasonic sensor mounted on the front of the car is used. This sensor is aimed toward objects in front of the chassis. All of these devices are connected to a predetermined WiFi location with the use of a Raspberry Pi 3 B+ module. The Raspberry Pi 3 B+ is equipped with dual-band 2.4/5GHz 802.11b/g/n/ac Wi-Fi allowing for the data transfer via WiFi [1]

**1.1.2 Project Materials**

This section includes a listing of the technologies and materials used to produce the proposed device, as follows:

* Raspberry Pi 3 B+
* Micro USB power cord
* SD-Card
* Portable power supply
* Ultrasonic sensor
* Temperature sensor
* Humidity sensor
* Raspberry Pi camera module
* 4WD Robot Chassis Kit
* AuviPal 5 megapixel Raspberry Pi Camera Module
* Computer monitor
* Working Computer
* REXQualis electronic kit (breadboard, jumper wires, resistors, etc)
* L298N dual H-bridge

****

**Figure 1: Physical Diagram of DAT CAR system [2]**

Figure 1 depicts a four-wheel-drive smart robot car chassis assembled kit that is used as the foundation for the DAT CAR system. The kit includes a PCB car chassis, four TT DC motors, four-car tires, two screw drives, motors wire, and a bundle of screws and nuts.

**1.2 2020 Project**

**1.2.1 Project Overview**

The objective is to integrate a cloud service to DAT CAR. The environmental data that is collected shall be saved to a cloud service, such as Microsoft’s OneDrive, Google Drive, or OwnCloud.

**1.2.2 Updated Purpose**

There has been a growing demand for data collection in remote environments. This includes environments that are unexplored, not accessible to humans, contain unknown conditions, or that are desired to be studied. Various environments may be too hot, cold, or contain radioactive residue deeming them inaccessible to humans. Environments may also contain unknown conditions, and they may be unexplored, which poses many safety concerns. Environmental conditions can be safely monitored for surface-based studies from a secure location using innovative technologies. The adopted solution for safe surface exploration is Data Acquisition Through a Car Altering Remote (DAT CAR), which is a rover that is designed to gain environmental knowledge in various areas. DAT CAR offers the ability for an individual to collect and visualize environmental data through a network connection. The user can control the rover from a remote location of their choice by having a Wi-Fi connection and the external Internet address of DAT CAR. Utilizing Flask, a micro web framework, the rover establishes a web server that the user is routed to when connecting to DAT CAR. From the Graphical User Interface (GUI), a video stream is shown of the environment in front of the rover for the user to visualize and evaluate the conditions of the surrounding area. Additionally, the GUI displays the temperature and humidity of the area and the distance of an object in front of the rover. The environmental data are relayed to a cloud server that DAT CAR will establish and connect to. The user can save environmental data and connect to the cloud server to study and evaluate the sets of data. DAT CAR was constructed through the research and study of software engineering, robotics, and networks. It was found that DAT CAR reflects innovative technologies for the use of safe exploration and the study of environments from a remote location.

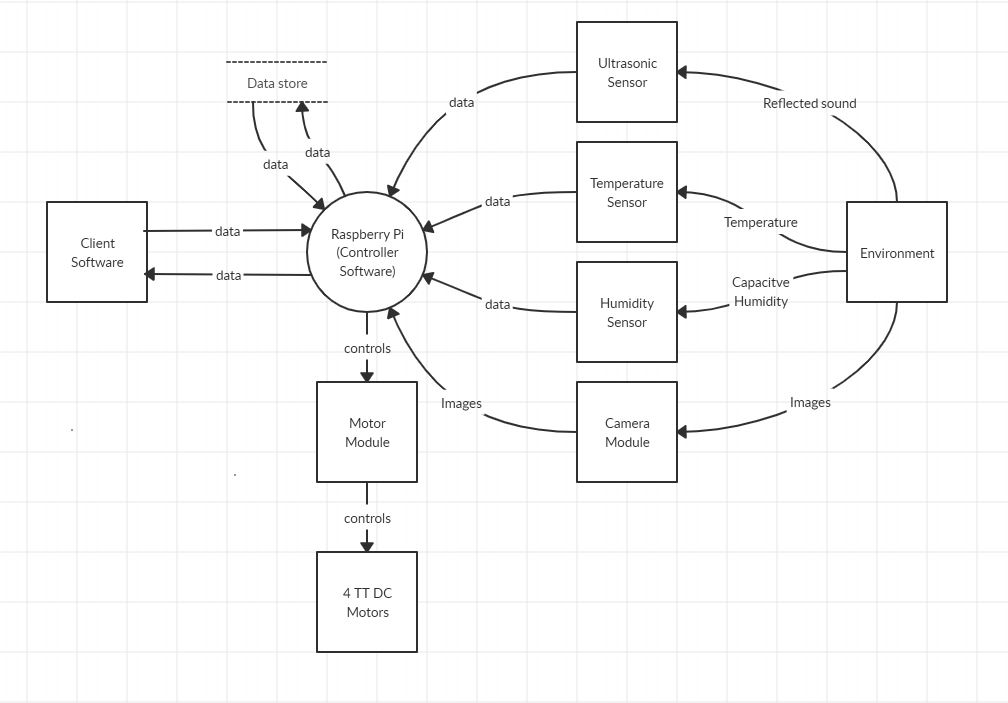
**2. Software Requirements Specification**

**2.1 2019 Project**

**2.1.1 Project Objectives**

The objective of this project is to create a fully functional DAT CAR device. The rest of this section outlines the project background and specifies software requirements.

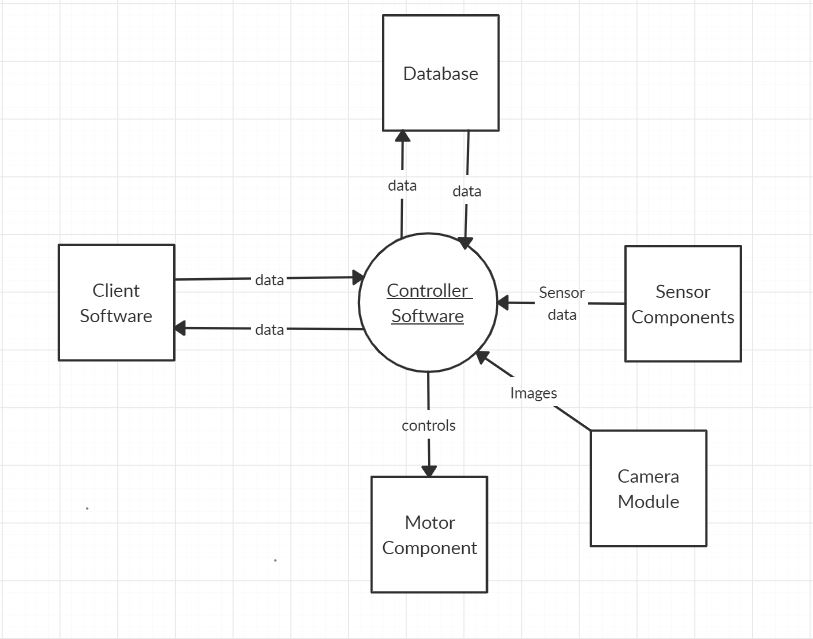
**2.1.2 Physical Diagram**



**Figure 2: Physical Diagram of DAT CAR**

The physical diagram in Figure 2 depicts the interaction with several devices in the DAT CAR. The DAT CAR central processing unit (CPU) utilizes a Raspberry Pi 3 B+ to interact with the end-user. The interaction with the end-user is in the form of displaying various information such as a live video feed, temperature, humidity, and distance of objects surrounding the RC car. The sensor devices connected to the Raspberry Pi process various information from the surrounding environment and send it to the Raspberry Pi controller. The controller sends the information through Wi-Fi to the end-user. In addition, the end-user can turn the RC car left or right and be able to move it forward or backward. The end-user is able to control the RC car with the use of input sent through Wi-Fi and ultimately to the motors of the car. The user can also save the current information being displayed on the GUI.

**2.1.3 Context Diagram**

****

**Figure 3: Context Diagram of the Remote Controlled Car**

**2.1.4 Software Requirements**

The software requirements are specified separately to the controller software and the client software

**2.1.4.1 Controller Software**

1. The controller software shall establish a webserver.
2. The controller software shall implement a GUI to the webserver that serves as the client software.
3. The controller software shall be able to save distance, temperature and humidity data into a text file.
4. The controller software shall be able to send distance, temperature, and humidity data to the client software.
5. The controller software shall utilize a camera and stream images to the client software.

**2.1.4.2 Client Software**

1. The client software shall be able to handle connection requests to the DAT CAR system through an IP address from a web browser.
2. The client software shall display the humidity and temperature of the RC car’s surrounding area to the end user’s monitor.
3. The client software shall display the distance of an object in front of the RC car to the end user’s monitor.
4. The client software shall be able to store current data being displayed from the system, such as distance, temperature, and humidity.
5. The client software shall be able to move the RC car left, right, forward, and backward by clicking the respective buttons on the GUI.

**2.1.4.3 Design Constraints**

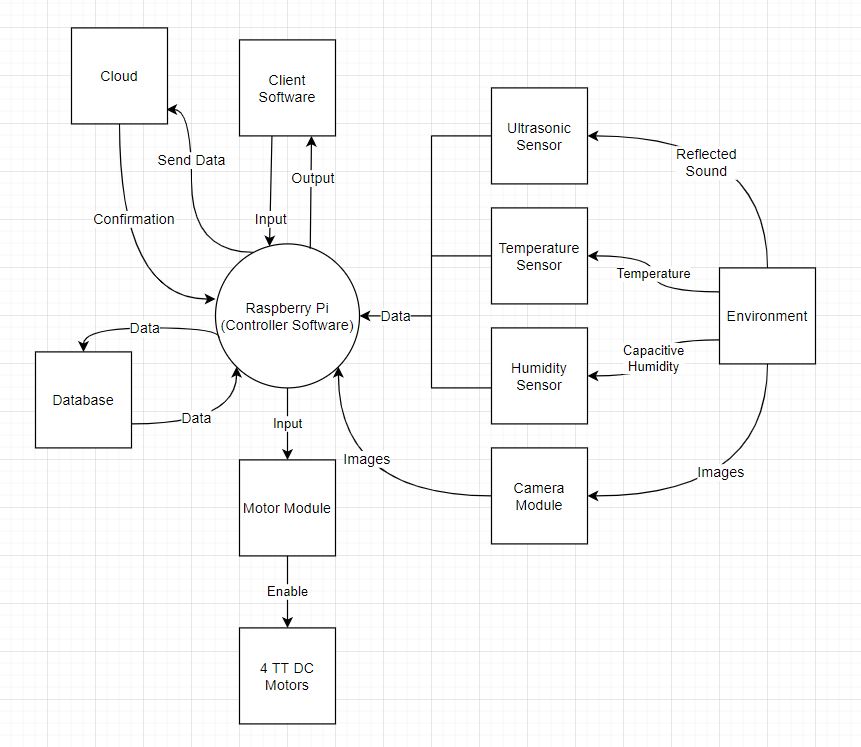
1. The DAT CAR system shall be able to connect to a predefined Wi-Fi endpoint.
2. The DAT CAR system shall utilize a temperature sensor and will be able to measure the temperature in the surrounding area.
3. The DAT CAR system shall utilize a humidity sensor and will be able to measure the humidity in the surrounding area.
4. The DAT CAR system shall utilize a camera and feed a stream to the user’s device.
5. The ultrasonic sensor shall be able to measure the distance from 2cm up to 450cm.
6. The DAT CAR system shall boot up in under 30 seconds after it is turned on.
7. The ultrasonic sensor shall be able to calculate distances within 2 seconds.
8. The robot chassis of the RC car will be able to handle a load up to 500g.
9. The response time of DAT CAR from user input shall be less than 5 seconds.

**2.2 2020 Project**

**2.2.1 Project Objectives**

The objective of this project is to create a fully functional DAT CAR device that transmits data to ownCloud. The rest of this section outlines the project background and specifies software requirements.

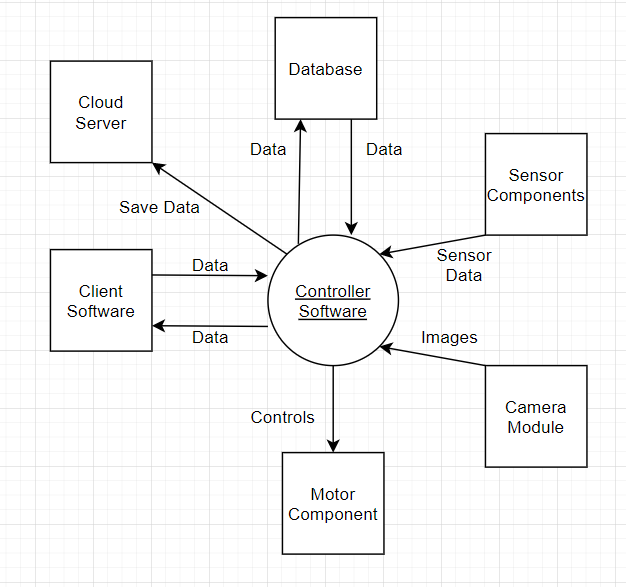
**2.2.2 Physical Diagram**



**Figure 4: DAT CAR Physical Diagram**

The physical diagram in Figure 4 depicts the interaction with several devices in the DAT CAR system. The DAT CAR central processing unit (CPU) utilizes a Raspberry Pi 3 B+ to interact with the end-user. The interaction with the end-user is in the form of displaying various information such as a live video feed, temperature, humidity, and distance of objects surrounding the rover. The sensor devices connected to the Raspberry Pi process various information from the surrounding environment and send it to the Raspberry Pi controller. The controller sends the information through Wi-Fi to the end-user. Furthermore, the user can also save the current information being displayed on the GUI to a cloud server that is established on DAT CAR. The end-user can also turn the rover car left or right and be able to move it forward or backward. The end-user is able to control the rover with the use of input sent through Wi-Fi and ultimately to the motors of the car.

**2.2.3 Context Diagram**

****

**Figure 5: DAT CAR Context Diagram**

Figure 5 depicts the new context diagram for DAT CAR. A cloud service was added as part of the objective of the maintenance project.

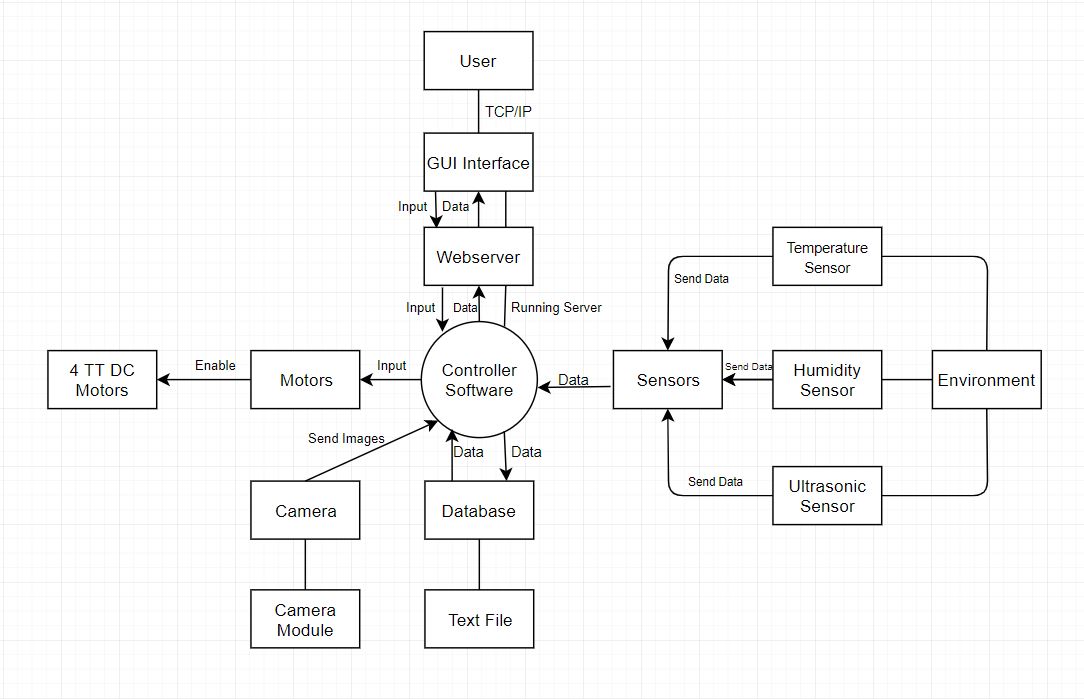
**2.2.4 Additional Requirements**

1. The controller software shall create an instance of an ownCloud server.
2. The controller software shall establish a connection to the cloud server.
3. The controller software shall relay the “saved data.txt” to the cloud server.
4. The controller software shall create a directory for the environmental data if one does not exist.
5. The controller software shall prompt the user that the environmental data was successfully relayed to the cloud server.
6. The controller software shall provide a link to the cloud server when the “saved data.txt” is relayed to the cloud server.

**3. Design Description**

**3.1 2019 Project**

**3.1.1 Software Architecture**

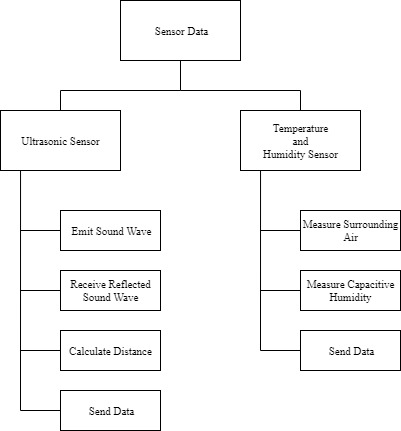
****

**Figure 6: Software Architecture of DAT CAR**

Figure 6 represents the Software Architecture for DAT CAR. The figure depicts how the data is transferred from the sensors to the database as well as displaying the data for the user on the GUI interface. The user also inputs the controls that interact with the motors and the database. The Controller Software (Raspberry Pi 3 B+ module) is the core of the entire system; it is transferring the data as well as giving power to the sensors in order to gather data. In addition, the controller software establishes a server that the end-user can connect to and view the GUI. There are five components to the software architecture, which are the sensors, motors, camera, database, and the webserver that the controller software establishes.

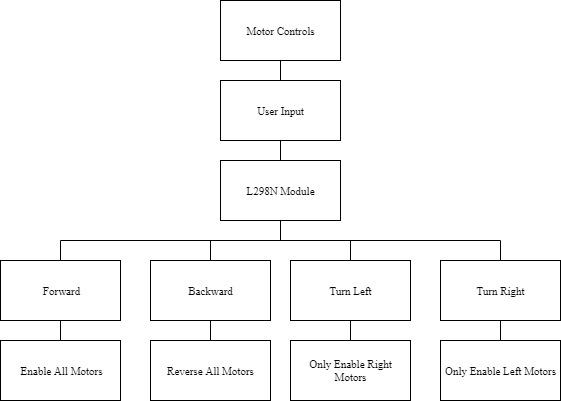
**3.1.2 Detailed Design**

**3.1.2.1 Software Structures**

****

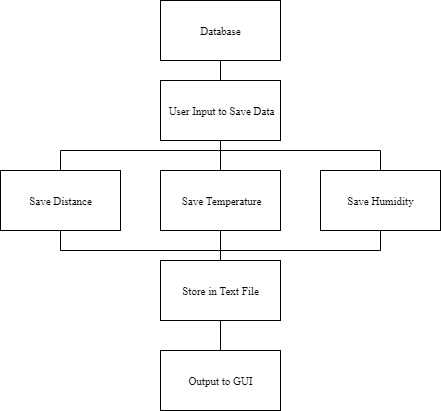
**Figure 7: Sensor Structure Chart**

Figure 7 depicts the structure of the sensors in DAT CAR. The ultrasonic sensor emits sound waves towards the object in front of it. The sound waves are reflected back at the ultrasonic sensor. Once the sound waves are received, the sensor calculates the distance of the object. The data is then sent to the controller software. The temperature and humidity sensor measures the surrounding air to calculate the temperature while also measuring the capacitive humidity to measure the humidity. The data is then sent to the controller software as well.



**Figure 8: Motor Controls Structure Chart**

Figure 8 depicts the structure of the motor controls of the DAT CAR. There are four TT DC motors that are connected to the wheels and controlled by the L298N module. The user interacts with the motors by inputting controls that are respective to the movements of DAT CAR. The four options of movement are forward, backward, turn left, and turn right. If the user inputs the forward command, then all of the left side motors of the chassis rotates counterclockwise, while the right side rotates clockwise, thus enabling a forward motion. If a backward movement is inputted, then the left side of the chassis rotates clockwise and the right side rotates counterclockwise to provide a backward motion. The turn left input enables the right-side wheels and rotates the chassis to the left. The right turn input enables the left side wheels and rotates the chassis to the right.



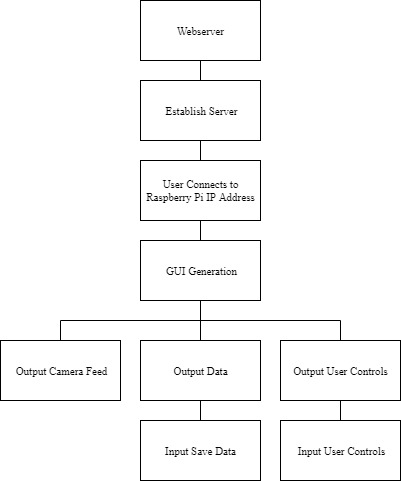
**Figure 9: Database Structure Chart**

Figure 9 depicts the structure of the database component of DAT CAR. The user can save the data that is displayed on the GUI by clicking the “Save Data” button. The current distance, temperature, and humidity data are then to be saved into a text file. The data that was saved is also shown on the GUI.



**Figure 10: Camera Module Structure Chart**

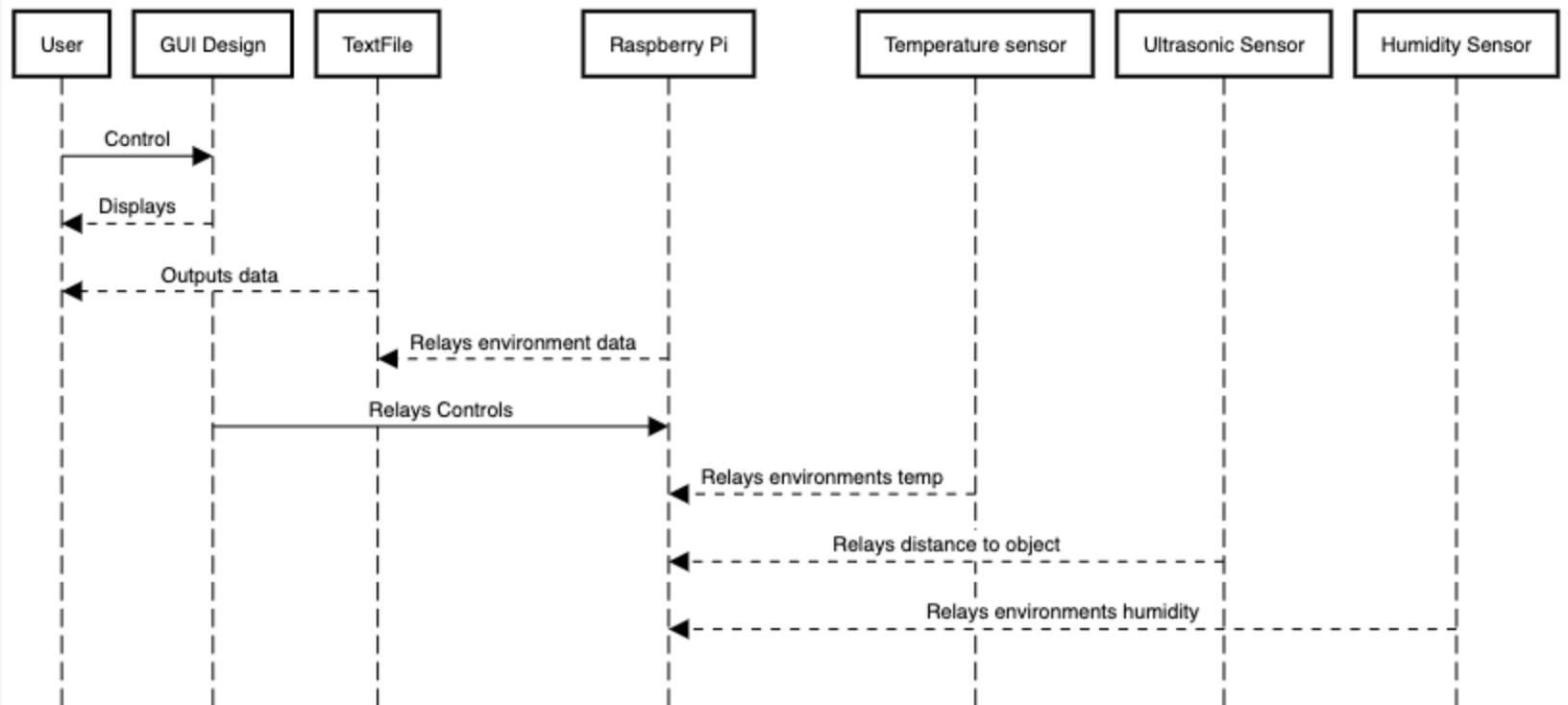
Figure 10 depicts the camera module structure in DAT CAR. The camera is enabled and takes images in succession to provide a video feed. The images capture are from the surrounding environment from DAT CAR. The video feed is sent to the controller software, which is then sent to the webserver. The web server displays the video feed on the GUI on the user’s client.



**Figure 11: Web Server Structure Chart**

Figure 11 depicts the structure of the web server component on DAT CAR. The controller software establishes a server. A user can connect to the server by inputting the Raspberry Pi’s IP address into a web browser. Once a user is connected to the Raspberry Pi, the GUI is generated for that user’s client. The GUI shows the video feed from the camera module, the output data from the sensors, and user controls of the motors on the RC car.

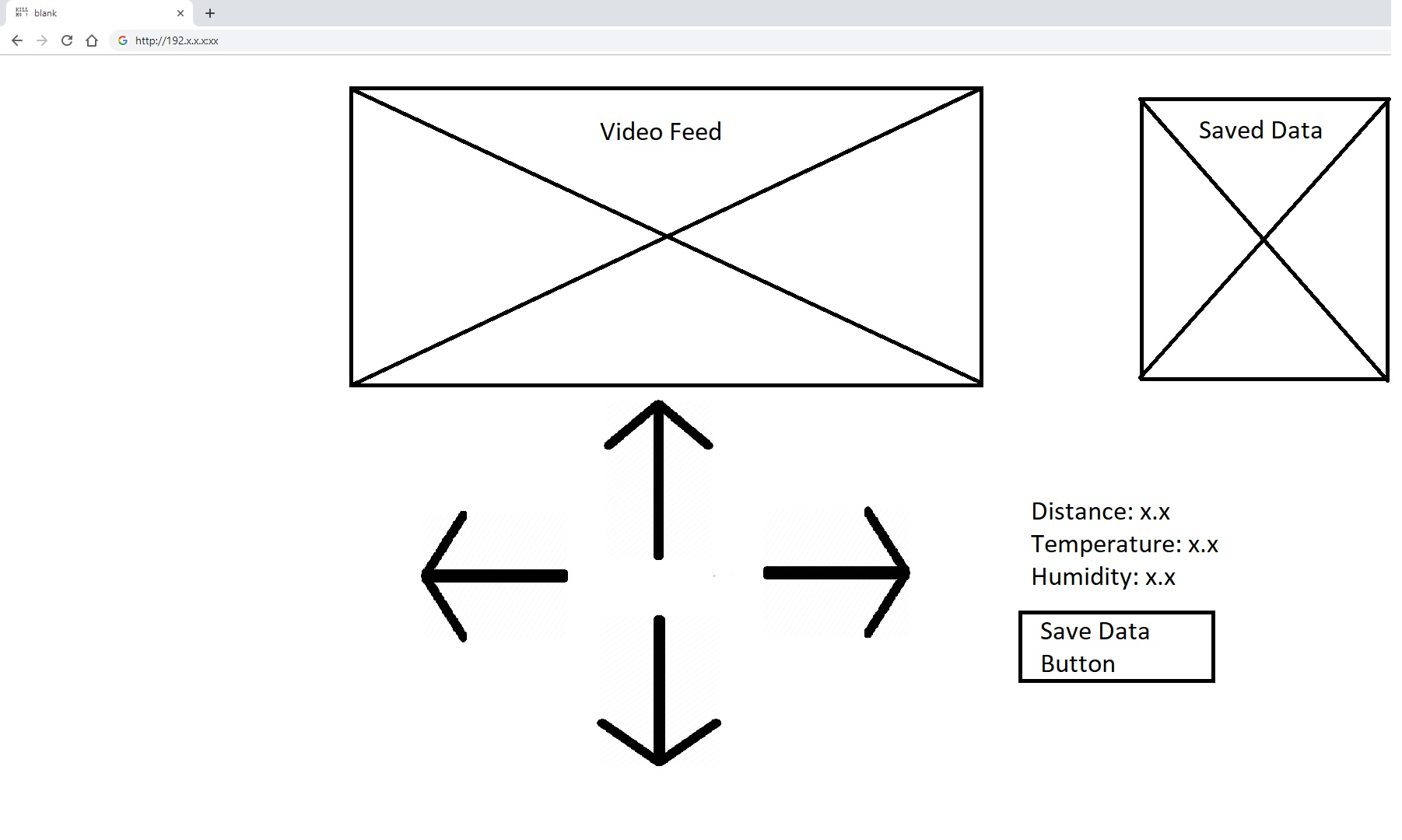
**3.1.3 Software Behavior**



**Figure 12: Sequence Diagram Displaying Software Behavior**

Figure 12 depicts the communication between the modules in a time sequence. The software communicates with each other constantly. The sensor modules relay their readings when the user sends the control through the GUI Design to the Raspberry Pi. This data is transferred via Wi-Fi to be saved to a text file that is viewable to the User.

**3.1.4 GUI Design**

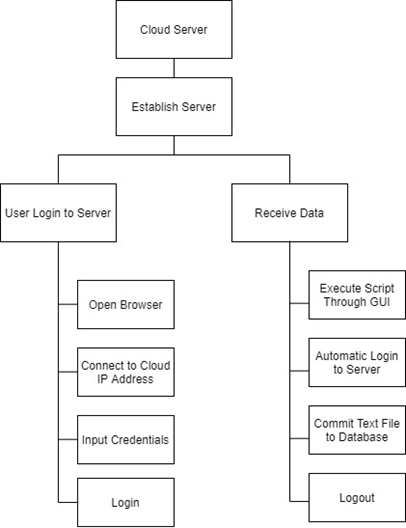


**Figure 13: Mockup GUI of User Application**

Figure 13 depicts the GUI Design for the project and how a user can control the DAT CAR via WiFi. A user can connect to the IP address of the microcontroller and this then displays the GUI design once connected. In Figure 11, the user can use the directional arrows in order to control the car. The GUI also contains a button that can save the current data of the environment; the system gathers the distance of an object in front of the car, as well as gather the temperature and the humidity of the environment surrounding the DAT CAR system.

**3.2 2020 Project**

The design for the maintenance portion of this product does not change too much from the previous design. The only thing changing is how the data is being saved to ownCloud rather than to a txt file. All cameras, sensors and GUI will stay the same and communicate the same as the 2019 version of the project. When the user selects the “Save Data Button”, this will save the current environment data to the ownCloud server rather than to a txt file.



**Figure 14: Cloud Server Structure Diagram**

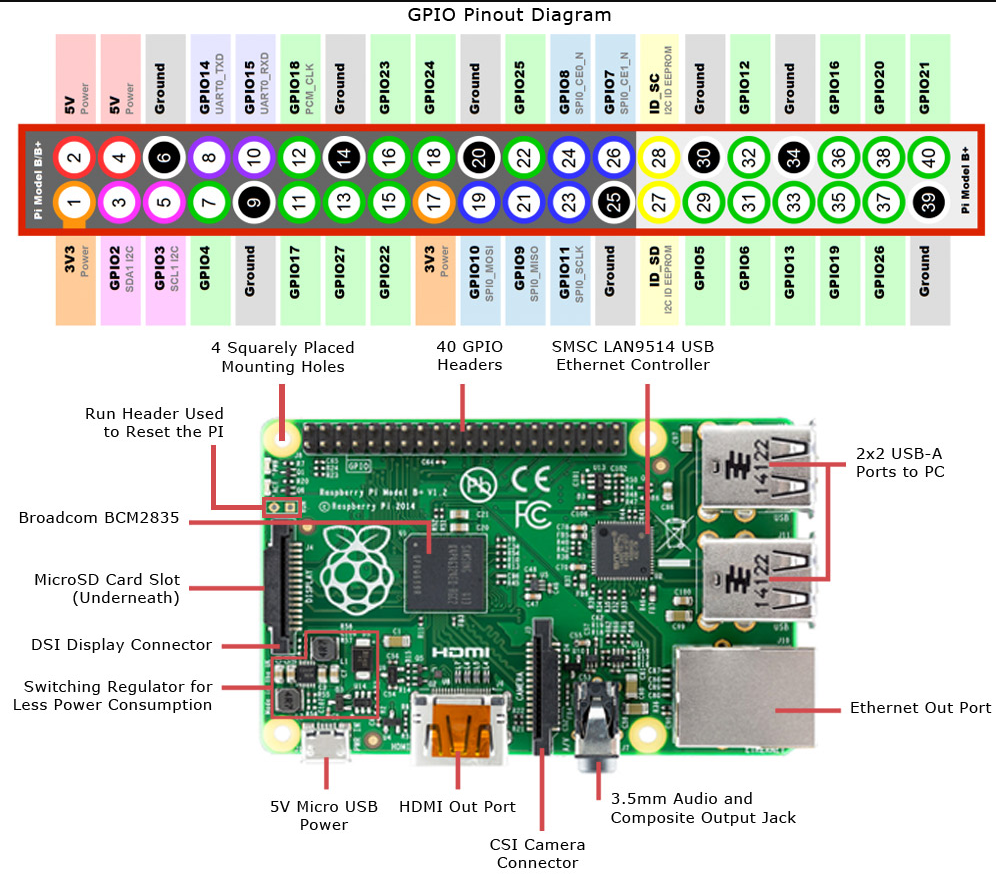
Figure 14 shows the cloud server structure, which was an added function for the Spring 2020 maintenance objective. A cloud server is established and interacts with two separate functions. The left path function is for a user to login to the cloud server with trailing steps on the process. A user would open an Internet browser and input the Raspberry Pi’s IP address followed by “/owncloud” to connect to the server. A login is prompted, in which a user would enter their credentials. The right path is a function for the server to receive data from DAT CAR. A python script is executed, which logs in to the cloud server and commits a text file to the cloud database, then logs out of the server.

**4. Implementation and Testing**

**4.1 2019 Project**

**4.1.1 Assembly and Coding**

**4.1.1.1 Raspberry Pi 3 B+**

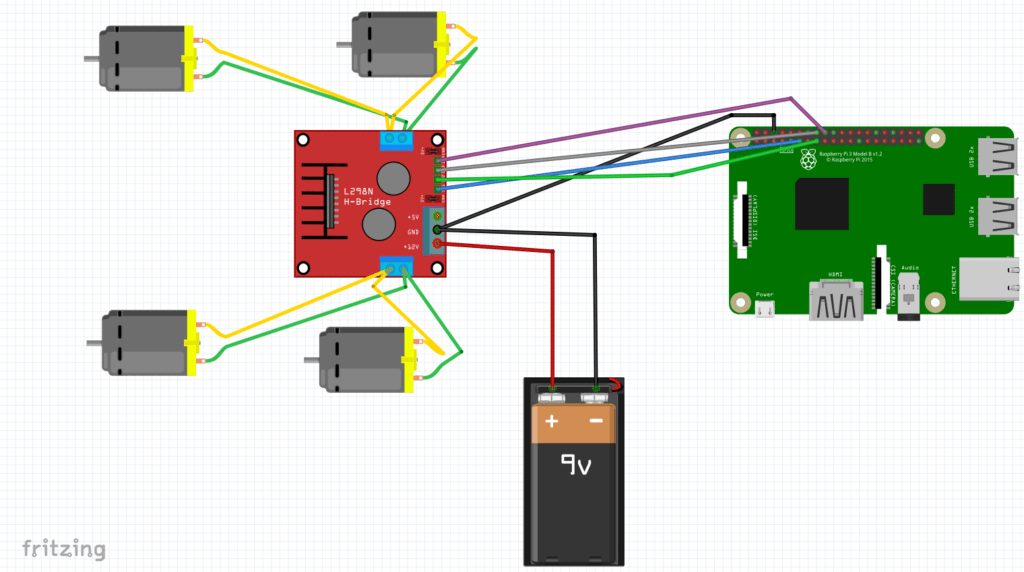


**Figure 15: Raspberry Pi Pinout Diagram [3]**

Figure 15 depicts the various input and output devices on the Raspberry Pi 3 B+, which is the controller software for the DAT CAR system. The Raspberry Pi is powered by a portable battery and connected to it with the 5V micro USB power port. Explicitly, this Raspberry Pi contains a single 40-pin expansion header with access to 28 general-purpose input/output (GPIO) pins. These pins are essential for the controller software. The ultrasonic, temperature and humidity sensors utilize the GPIO pins to gain power and transmit data to the Raspberry Pi. Utilizing a breadboard was done to connect the sensors to the Raspberry Pi as well. In addition, the GPIO pins power the motor module that is used to control the movements of the four TT DC motors and move the RC car chassis. A camera module is also used in this project and is connected to the CSI camera connector port to allow a live video feed to be sent to the end-user.

Many tutorials can be found online for setting up Raspbian OS. Raspbian is the OS that DAT CAR runs on. A tutorial that may be used is: <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up>

**4.1.1.2 L298N Dual H-Bridge Module**

****

**Figure 16: Circuit Diagram for Motor Module [4]**

Figure 16 depicts how the four TT DC gear motors gain power via the L298N Dual H-Bridge Module. The motor module is powered by a 9v battery. Each of the TT DC motors is connected to the motor module with a power and grounding wire. The motor module is connected to the Raspberry Pi through the GPIO pins at port 17, 22, 23, and 24 along with a grounding pin. GPIO programming is used to control the motors by activating the input and output ports on the L298N module. This Dual H-Bridge module has a separate battery source from the Raspberry Pi. Separate batteries are used as a precaution to back EMF. This event can occur when any inductor, the motor coil in this instance, is switched off. The motor coil can produce a huge voltage spike and may terminate a control circuit.

m11=17

m12=22

m21=23

m22=24

GPIO.setmode(GPIO.BCM)

GPIO.setup(m11, GPIO.OUT)

GPIO.setup(m12, GPIO.OUT)

GPIO.setup(m21, GPIO.OUT)

GPIO.setup(m22, GPIO.OUT)

GPIO.output(m11 , 0)

GPIO.output(m12 , 0)

GPIO.output(m21, 0)

GPIO.output(m22, 0)

@app.route('/turn\_left')

def left\_side():

data1="LEFT"

GPIO.output(m11 , 0)

GPIO.output(m12 , 0)

GPIO.output(m21 , 1)

GPIO.output(m22 , 0)

return 'true'

@app.route('/turn\_right')

def right\_side():

data1="RIGHT"

GPIO.output(m11 , 1)

GPIO.output(m12 , 0)

GPIO.output(m21 , 0)

GPIO.output(m22 , 0)

return 'true'

@app.route('/go\_forward')

def up\_side():

data1="FORWARD"

GPIO.output(m11 , 1)

GPIO.output(m12 , 0)

GPIO.output(m21 , 1)

GPIO.output(m22 , 0)

return 'true'

@app.route('/go\_backward')

def down\_side():

data1="BACK"

GPIO.output(m11 , 0)

GPIO.output(m12 , 1)

GPIO.output(m21 , 0)

GPIO.output(m22 , 1)

return 'true'

@app.route('/stop')

def stop():

data1="STOP"

GPIO.output(m11 , 0)

GPIO.output(m12 , 0)

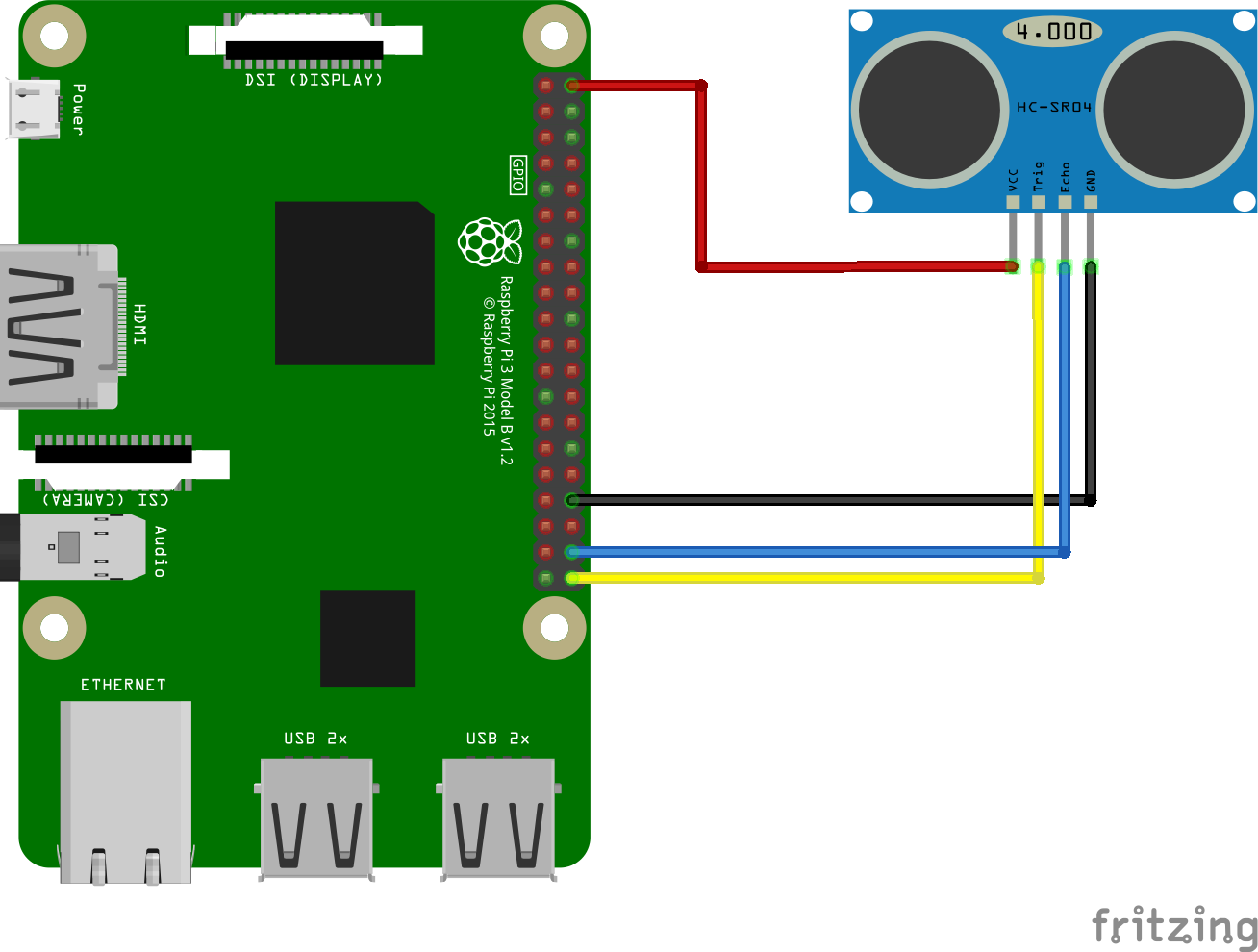
GPIO.output(m21 , 0)

GPIO.output(m22 , 0)

return 'true'

The Python code snippet depicts the setup of the GPIO pins for the motor module, as well as the functions defining respective movements of the chassis. The variables m11, m12, m21, and m22 are assigned numbers that are associated with the pin number on the Raspberry Pi. The GPIO.setmode (GPIO.BCM) is referring to the pins by the “Broadcom SOC channel” number. The numbers can be seen after “GPIO” in the green rectangles in Figure 12. The GPIO.setup section declares the “pin modes” that are being used for the motor module. The GPIO output section writes the pin values as the boolean value 0, or false. The @app.route that are before defined functions is a built-in wrapper in Flask for generating routes. The routes are referring to the HTML file that is used in the project. When a respective button is clicked on the GUI, the HTML code calls the function that is associated with the specific route. There are five functions defined that are responsible for executing the movements of the chassis. The five functions defined are moving forward, backward, left, right, and stop. Each of the functions sets the output of the motor module to either false or true and depending on the output, the four TT DC motors of the wheels will either be on or off and are either moving clockwise or counterclockwise.

**4.1.1.3 Ultrasonic Sensor**

****

**Figure 17: Circuit Diagram for Ultrasonic Sensor [5]**

Figure 17 depicts how the Ultrasonic Sensor was connected to the Raspberry Pi. An HC-SR04 was used for this project. A breadboard was utilized with reference to the diagram to connect the sensor to the Raspberry Pi. Ports 21 and 20 are connected to the sensor along with the 5v power and ground pins. With the use of a breadboard, it could be successfully mounted to the chassis with the sensor pointing out in front of the car. The ultrasonic sensor emits soundwaves that reflect on objects in front of it. The reflected sound is calculated by the sensor to find the distance of an object in front of it. The sensor communicates to the Raspberry Pi module via GPIO pins, thus allowing the module to read and write. The data is then transferred to the Raspberry Pi. The Raspberry Pi sends the data from the sensor to the GUI interface that is established on the server.

TRIG = 21

ECHO = 20

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

def findDistance():

GPIO.output(TRIG, False)

print ("Waiting For Sensor To Settle")

time.sleep(2)

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:

pulse\_start = time.time()

while GPIO.input(ECHO)==1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

global distance

distance = pulse\_duration \* 17150

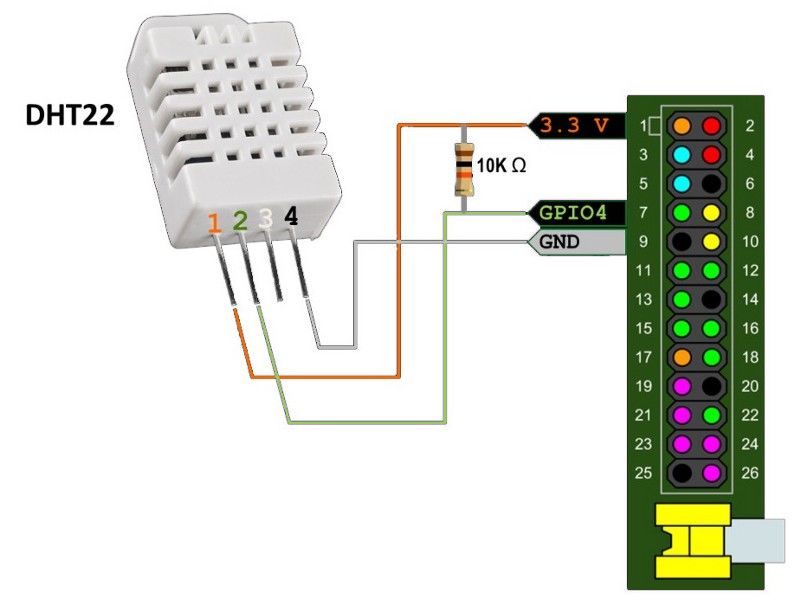
distance = round(distance, 2)

print ("Distance:",distance,"cm")

return 'true'

The Python code snippet depicts the setup of the GPIO pins for the ultrasonic sensor, as well as the function that calculates the distance. The variables TRIG and ECHO are assigned numbers that are associated with the pin number on the raspberry pi. The GPIO.setmode(GPIO.BCM) is referring to the pins by the “Broadcom SOC channel” number. The numbers are after “GPIO” in the green rectangles in Figure 12. The GPIO.setup section declares the “pin modes” that are being used for the ultrasonic sensor. The GPIO output section writes the pin values as 0, or false. The “findDistance()” function uses both a trigger (TRIG) and an echo. The trigger pin is used to trigger the ultrasonic sound pulses, while the echo pin produces a pulse when the reflected signal is received. The length of the pulse is proportional to the time it took for the transmitted signal to be detected [8]. The function calculates the distance between the two variables.

**4.1.1.4 Temperature and Humidity Sensor**



**Figure 18: Temperature and Humidity Sensor Circuit Diagram [6]**

Figure 18 displays how the temperature and humidity module is connected to the Raspberry Pi GPIO pins. A DHT22 was used for this project. A breadboard was used to connect the temperature and humidity sensor to the Raspberry Pi with reference to the diagram. GPIO pin 4 is used along with the 3.3v power and grounding pins. The sensor module detects the temperature of the environment as well as the capacitive humidity of the environment. The sensor module contains four pins. Pin 1 is for the Power supply. Pin 2 is for Data collection. Pin 3 is not to be connected to the Raspberry Pi module. The last one, Pin 4 is the grounding pin. The sensor also utilizes a 10k ohm resistor to control the flow of the current and limit it.

import Adafruit\_DHT

DHT\_SENSOR = Adafruit\_DHT.DHT22

DHT\_PIN = 4

GPIO.setmode(GPIO.BCM)

def findTempHumi():

humidity, temperature = Adafruit\_DHT.read\_retry(DHT\_SENSOR,

DHT\_PIN)

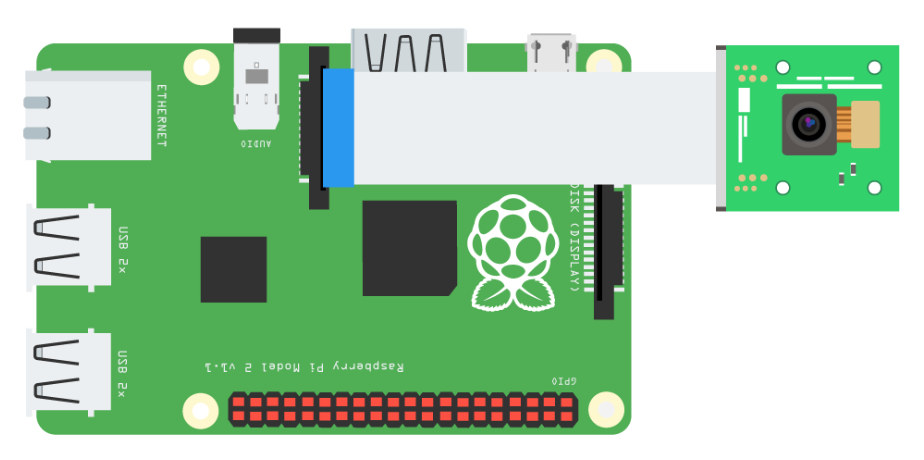
global tempHumi

tempHumi="Temp={0:0.1f}\*CHumidity={1:0.1f}%\n".format(temperature,humidity)

return 'true'

The Python code snippet depicts the setup of the GPIO pins for the temperature and humidity sensor, as well as the function that calculates the temperature and humidity. The function also formats the output of the temperature and humidity. The code imports the Adafruit\_DHT library. The variable DHT\_Sensor is assigned a value that is associated with the DHT22 sensor in the Adafruit\_DHT library. The variable DHT\_PIN is assigned the number that is associated with the pin that it is plugged into on Raspberry Pi’s GPIO board. The GPIO.setmode(GPIO.BCM) is referring to the pins by the “Broadcom SOC channel” number. The numbers are after “GPIO” in the green rectangles in Figure 12. The imported library contains the code for the DHT22 sensor. The temperature and humidity variables are assigned a value that is found by calling a function within the Adafruit\_DHT library that calculates the temperature and humidity. The calculated values are then formatted in a new variable called “tempHumi”.

**4.1.1.5 Raspberry Pi Camera Module**

****

**Figure 19: Camera Module Diagram [7]**

Figure 19 shows a camera module inserted into the CSI camera connector port. Python is used for photo imaging within this project. The python code allows for the server to turn on the camera module and stream it to the user application. The camera is utilized so that the user can see where the RC Car is located and can use the controls appropriately. Depending on the internet connection, the latency of the camera should be less than five seconds. If a user is connected to the same network, then the latency is less than four seconds. If the internet speed has at least 5mb download speeds and 5mb upload speeds, then the latency is less than one second.

from camera\_pi import Camera

def gen(camera):

while True:

frame = camera.get\_frame()

yield (b'--frame\r\n'

b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

#Stream the video to the html file

@app.route('/video\_feed')

def video\_feed():

return Response(gen(Camera()),

mimetype='multipart/x-mixed-replace; boundary=frame')

The code snippet depicts the camera being created and streamed to the HTML file. The code imports the file camera\_pi.py by Marcelo Rovai [8]. The camera\_pi.py code establishes a class named Camera with an object parameter. Within the code, it initializes the camera, gets the frames, sets the resolution, and continuously captures new frames. In the code snippet, the first function generates a camera object and retrieves the frames. The second function, “def video\_feed()” streams the video through a Flask application route, where it is the source of an image in the HTML code.

**4.1.1.6 Database**

@app.route('/save\_data')

def saveSensorData():

findTempHumi()

findDistance()

f = open("saved data.txt", "a+")

f.write("Distance was %d" % distance + "cm\n")

f.write(tempHumi + "\n")

f.close()

return 'true'

The database that is used in the project is a text file. The code snippet routes the “saveSensorData” function when a certain button is clicked on the GUI. Within the function, the “findTempHumi” function is called that calculates the current temperature and humidity. The “findDistance” function is also called that calculates the distance of an object in front of the ultrasonic sensor. Within the directory of the python script, the code opens a text file named “saved data.txt”, and if it does not exist, then it creates the file. It is defined in the parameters of line five that it appends information to the file, so no information already in the file is overwritten. The distance value is appended to the text file followed by the temperature and humidity values. The file is closed and the function is done executing.

**4.1.1.7 Web Server**

from flask import Flask

from flask import render\_template, request

from flask import Response

app = Flask(\_\_name\_\_)

def index():

findDistance()

findTempHumi()

return render\_template('testWeb.html', dist=distance, temp=tempHumi)

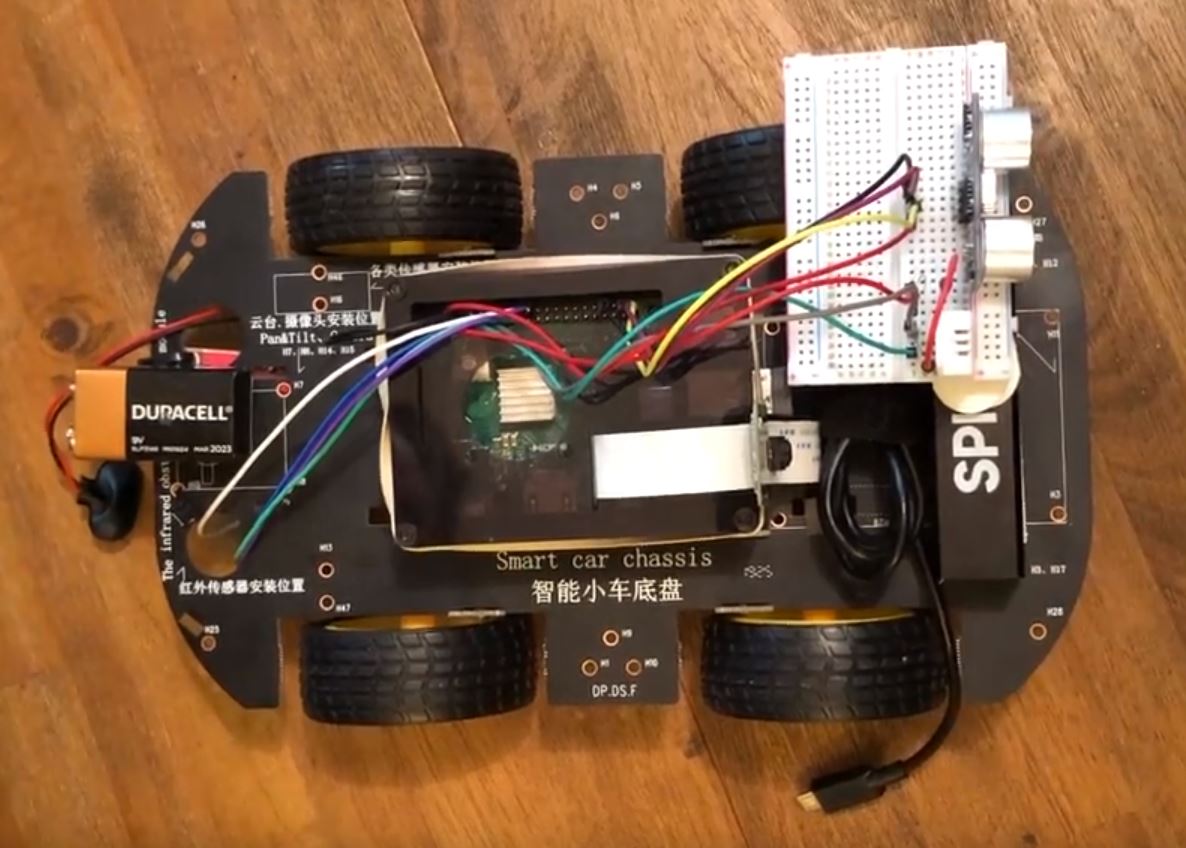
if \_\_name\_\_ == "\_\_main\_\_":

print ("Start")

app.run(host='0.0.0.0',port=5010, debug=True, threaded=True)

The web server that is established with the controller software utilizes Flask. The code snippet imports the Flask files so a web server can be established. A variable named “app” is declared and instantiated with a Flask object. The function “index” is used and renders an HTML template. In addition, the function calls “findDistance” and “findTempHumi” functions to pass variables to the HTML code. The variables contain sensor data that are displayed on the GUI. At the end of the code snippet, a web server is being established. The IP address that is set is the local host’s IP address. The port number is 5010. With the IP address and port number, a user can connect to the webserver by inputting the IP address followed by a colon then the port number. As a result of Flask, other functions can be routed to the webserver.

**4.1.1.8 Final Assembly**

****

**Figure 20: Complete DAT CAR Assembly**

Figure 19 shows the final assembly of the DAT CAR. The housing of the Raspberry Pi is attached to the top of the chassis with the use of velcro strips. The power supply for the Raspberry Pi is also attached near the front of the chassis where it is used as a fastening point for the sensors connected to the breadboard. Near the rear of the chassis is the 9v battery that is attached to the motor module. The motor module is underneath the chassis and is attached with zip ties. The wiring to each component is done with respect to their diagrams from previous sections. A rubber band is wrapped around the case of the Raspberry Pi and the ribbon for the camera module. This method allows the camera to be upright and pointed to the front of the chassis.

**4.1.2 Testing**

|  |  |
| --- | --- |
| Test Case | 1 |
| Requirement | 2.4.2 - 5: The client software shall be able to move the RC car left, right, forward, and backward by clicking the respective buttons on the GUI. |
| Input | User clicks left, right, forward, and backward buttons on the GUI |
| Output | DAT CAR correctly moves into the respective direction that is selected on the GUI. |

Post-test discussion:

The specific requirement has been met with test case 1. However, in testing, it was found that when a user presses a button and drags the pointer off of the button then releases the button, the button continuously registers itself as being pressed. DAT CAR continuously moves in the direction of the bugged button until a button is pressed and depressed with the cursor over the button. This discrepancy is prompted because the user may not want the chassis to move in the direction of the bugged button. A suggestion for the user is to always have the cursor over the button while it is being pressed and depressed. A suggestion for the development team is to add Javascript code that depresses the button when the cursor is no longer over the button.

|  |  |
| --- | --- |
| Test Case | 2 |
| Requirement | 2.4.1 - 3: The controller software shall be able to save distance, temperature and humidity data into a text file. |
| Input | The user selects the “Save Data” button on the GUI which calls the controller software to save distance, temperature, and humidity values into a text file. |
| Output | The controller software calls the distance, temperature and humidity functions then appends the values to a text file named “saved data” within the folder where the source code is located. If the file named saved data does not exist, then the controller software creates the text file. |

Post-test discussion:

The specific requirement has been met with test case 2. There is no discrepancy between the requirement and actual results.



**Figure 21: Sensor Data in Text File**

Figure 21 shows the appended sensor data in a text file. The distance, temperature, and humidity values were correctly stored in the text file.

|  |  |
| --- | --- |
| Test Case | 3 |
| Requirement | 2.4.3 - 6: The DAT CAR system shall boot up in under 30 seconds after it is turned on. |
| Input | The controller software (Raspberry Pi) is turned on by a portable power supply. |
| Output | The controller software script is running on the bootup of the raspberry pi. |

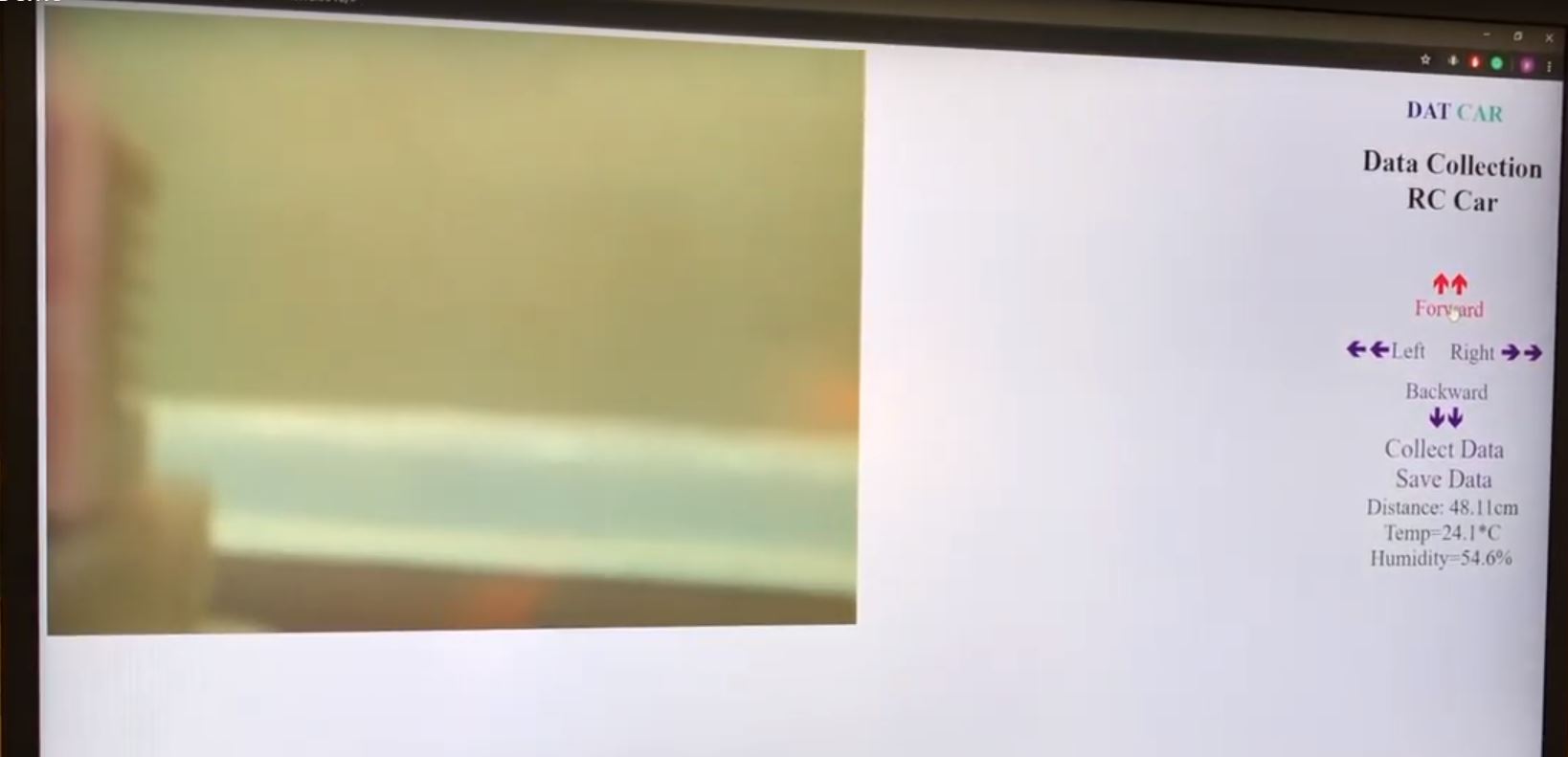
Post-test discussion:

The specific requirement has not been met with test case 3. There is a discrepancy between the requirement and the results. After DAT CAR is turned on, it took around 60 seconds to successfully run the controller software scripts. There are three suggestions for the discrepancy. The first suggestion would be to add a physical button on DAT CAR that can run the script to start the controller software, which should boot up almost immediately if the Raspberry Pi is already on. The second suggestion is to upgrade the Raspberry Pi, therefore decreasing bootup times. The third suggestion is to edit the requirement with post knowledge on the software.

|  |  |
| --- | --- |
| Test Case | 4 |
| Requirement | 2.4.1 - 5: The controller software shall utilize a camera and stream images to the client software. |
| Input | A Raspberry Pi camera module is used. A camera object is created and images are taken from the Raspberry Pi camera module that are streamed to the client software GUI |
| Output | The client software GUI receives the images and updates the source when a new one is received. |

Post-test discussion:

The specific requirement has been met with test case 4. There is no discrepancy between the requirement and the results. The time in which the source image is updated is dependant on the internet upload and connection speeds on the controller and client software.

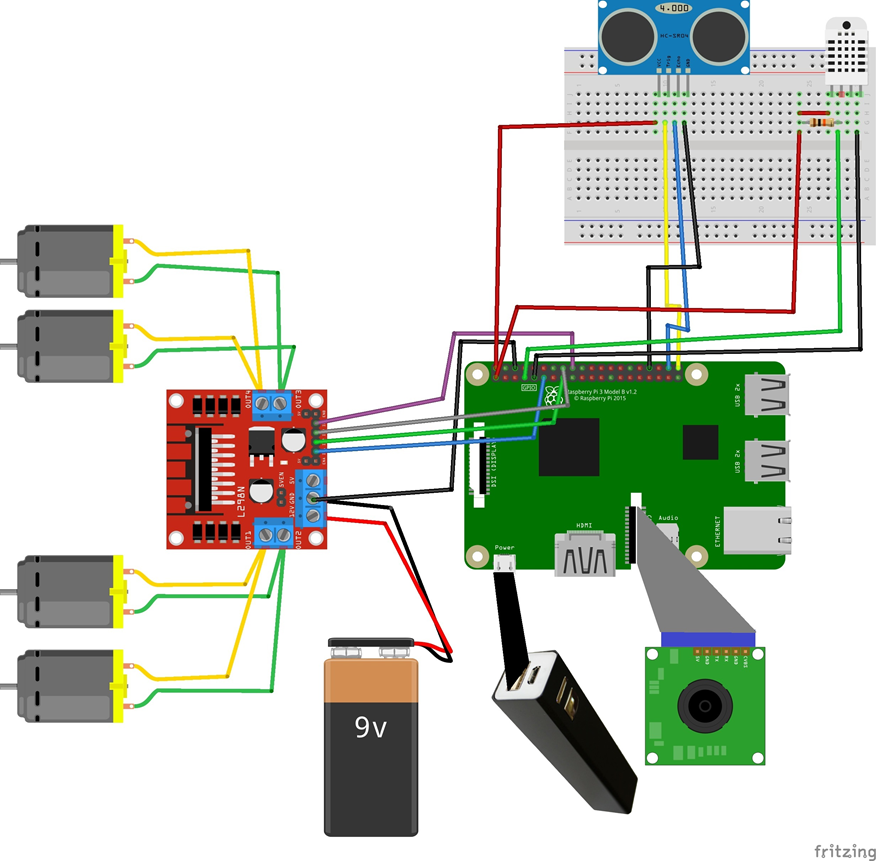


**Figure 22: GUI for DAT CAR**

Figure 22 shows the GUI for the client software. The image on the left side of the GUI is taken from the camera module. The camera is taking multiple images and the controller software is streaming the images to the client software GUI. The source image in the HTML file is updated, therefore streaming images and creating a video stream.

**4.2 2020 Project**

**4.2.1 Assembly and Coding**



**Figure 23: DAT CAR Circuit Diagram**

Figure 23 depicts the final circuitry for the physical hardware composing DAT CAR. The four DC TT biaxial motors are connected to the L298N Dual H-Bridge with positive and negative leads. The motor module is powered by a 9v battery with a snap connector. The motor module is connected to the Raspberry Pi’s GPIO ports. The Pi is powered by a power supply stick. The Pi uses a Camera Module V2-8 Megapixel, 1080p camera, which is connected directly to it. A bread board is utilized to connect the Ultrasonic sensor and the DHT22 sensor.

**4.2.2 ownCloud Server**

Instructions for installing OwnCloud to DAT CAR can be found at <https://www.avoiderrors.com/owncloud-10-raspberry-pi-3-raspbian-stretch/>**.** The database that the cloud server on DAT CAR utilizes is an SQL one. The text file containing the environmental data is sent and saved to the cloud server’s database. After the cloud server is successfully installed on the rover, it should automatically boot up when the DAT CAR is turned on. Within a web browser’s search bar, type “http://<IPofRaspberryPi>/owncloud” but replace “IPofRaspberryPi” with the actual IP of the Pi. The initial setup of administrative controls should prompt for user input. Instructions for obtaining the Raspberry Pi’s IP address can be found in section 3.2.

Furthermore, the “pyocclient” should be installed to make it possible to connect to an ownCloud instance and perform python commands. Pyocclient is a library that provides functions to abstract away HTTP calls for various ownCloud APIs. Instructions to installing the library can be found at: <https://github.com/owncloud/pyocclient>

def saveToCloud():

oc = owncloud.Client('http://0.0.0.0/owncloud')

oc.login('username', 'password')

oc.put\_file('testdir/remotefile.txt', 'saved data.txt')

link\_info = oc.share\_file\_with\_link('testdir/remotefile.txt')

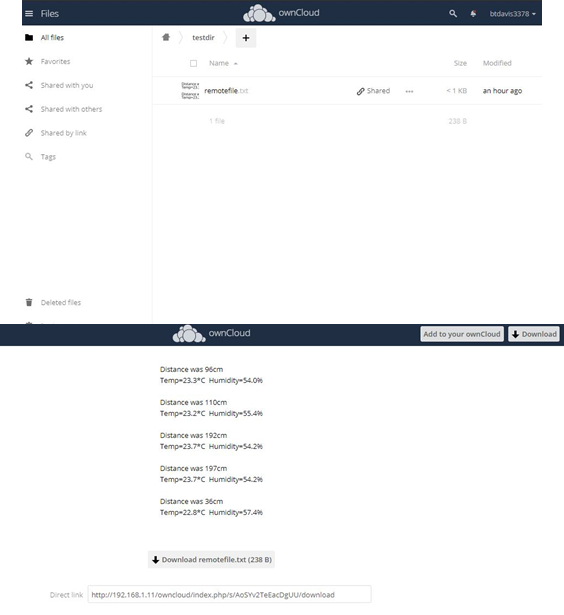
The code snippet above can be found within the “main.py” script. Open the script with a preferred IDE/text editor. Geany was used for the editing of the script. Line 2 of the code snippet should be edited to reflect the IP address of the Raspberry Pi, and line 3 should reflect the credentials of the login created to the cloud server. The first parameter of line 4 should reflect the path that the saved environmental data shall be saved in. The last line within the code snippet should also be altered to reflect the same path from line 4.

**4.2.3 Testing**

|  |  |
| --- | --- |
| Test Case | 5 |
| Requirement | 2.2.4 - 7 - The controller software shall relay the “saved data.txt” to the cloud server. |
| Input | Data values: Distance, Temperature, and Humidity of the environment |
| Output | .txt file sent to ownCloud server directory |

Post-test discussion:

The requirement has been met with test case 5. There is no discrepancy between the data values of the environment and the data values being saved within an ownCloud server. The time in which the data becomes available will depend upon the user’s computer and connection.



**Figure 24: .txt File Saving on ownCloud Server**

Figure 24 depicts the data being saved within ownCloud. The data was successfully relayed from the controller software to the cloud server. The data is saved as a .txt file and then when the .txt file is opened, the user can view data that was previously saved. The file is overwritten every time it is relayed, but the contents are always updated. This means that previous data within the text file are not erased.

**5. Operating Instructions**

**5.1 First Time Setup**

The source code of DAT CAR can be cloned from the Github repository: <https://github.com/btdavis3378/DAT-CAR>. The repository can be cloned into a directory of the user’s choice. When the user clones the DAT CAR repository, they will need to add the “Main.py” script to the boot up configuration. Completing this will allow for the DAT CAR system to boot up on its own for future uses. Adding the script to the boot up configuration can be done by:

1. Locating the directory path of the main.py file.
2. Open the terminal and run “sudo nano /etc/rc.local”.
3. Add “python3 <path of main.py file>” just above the exit 0 in that file.
4. Press Ctrl+X and save the file.

Restart the Raspberry Pi device to ensure that the device boots up on its own with the new script. The terminal command, “ps -aef | grep python3” can be run to ensure that the script is running successfully.

**5.2 Acquiring Raspberry Pi IP Address**

The IP address of the Raspberry Pi can be obtained in a number of ways. For this tutorial, two methods are listed to obtain the IP address, but other ways of finding the IP address can be found online.

1. First Method: If the Raspberry Pi is connected to a monitor, keyboard, and mouse, then open the terminal window. Type “ifconfig” in the terminal and hit enter. Under “wlan0” there is a row that contains similar contents to, “inet x.x.x.x”. The series of four numbers is the internal IP address of the Raspberry Pi connected to the local network. Take note of the IP address.
2. Second Method: The IP address of the Raspberry Pi can also be found by doing what's known as an "IP Scan" across a network. Most home networks will have about 254 possible combinations of IP addresses, so looking at them all is an easy task for a desktop or laptop computer. A program called Angry IP Scanner ( http://angryip.org/ ) is recommended to scan the network. This program runs on Linux, macOS X or Windows and lets an individual scan an entire local network to find devices, including any Raspberry Pi's that are running. When the Angry IP Scanner program is executed it should automatically pick a sensible IP range for all the devices on the network. Find the IP address of the Raspberry Pi and take note of it.

**5.3 Connecting to DAT CAR**

Open an internet browser, and in the search bar type “http://<IPAddressofPi>:5010” and hit enter. The browser should connect to the webserver that the Raspberry Pi is hosting, and the GUI should display.

**5.4 Controls**

When the GUI is displayed for a user, they will see five main things on this page. The name of the product, directional buttons, the collect and save data buttons, the data itself, and then the live feed pointing in front of the DAT CAR system. The name of the product will be in the top right and will not be able to be manipulated by a user. Beneath the name of the product, the user will see the directional buttons for the system. They will have a “Forward” button, “Left” button, “Right” button, and “Backward” button. The buttons functionality is how it sounds, this is how users will control the DAT CAR system via a mouse. **Warning:** You can only press one direction at a time. Beneath the controls and in the bottom right of the GUI display the user will see the data acquiring buttons. There are two buttons, “Collect Data” and “Save Data”. “Collect Data” will display the current environmental data in the data section of the GUI while the “Save Data” button will take the current data on screen and save it to a text file for local use and this text file will be sent to the cloud server. The cloud server can be accessed via a web browser using the Raspberry Pi IP address followed by “/owncloud/”. The following section the user will be displayed is the data section, this will show three values in a list format; distance, temperature, and humidity of the surrounding environment. The last section of the GUI display will be the live feed section, this will show the view in front of the car while also showing in which direction the sensors are facing.

**5.5 Cloud Server Connection**

In an Internet browser, type “http://<IPAddressofPi>/owncloud”. During the installation of the cloud server, login credentials should have been established. The server should prompt for the login credentials. After inputting the username and password, hit enter. The home page of the cloud server can be seen, and a folder titled “DAT CAR Data” should be displayed along with other directories. Within the folder, a document containing the saved data from DAT CAR’s sensor can be found. The document can be downloaded and further observed.

**5.6 Port Forwarding**

To connect to the Raspberry Pi over the Internet on a different network, port forwarding must be set up on the Raspberry Pi’s network. The configuration of the router should be changed to forward all inbound traffic from the internet on a specific port to the local IP address of the Raspberry Pi. Most routers have the feature to port forward, but the way of doing so will vary from router to router. Consult the router’s user manual for instructions. Additionally, there are alternative third-party services rather than port forwarding that are available.

**Warning:**

Port forwarding exposes a network port on your private LAN to the public internet. This is a known security vulnerability and must be managed carefully.

**6. Conclusion**

**6.1 2019 Project**

DAT CAR was able to successfully collect data based on the environment’s distance, temperature, and humidity with the use of web controls that activated the sensors and manipulated the movement of the chassis. There are five main components to the software architecture that define DAT CAR, which are the sensors, motors, camera, database, and the webserver.

The sensors that were successfully implemented on the DAT CAR system were: an ultrasonic sensor and a temperature with a humidity sensor. All the sensors are able to accurately collect data on the environment surrounding the DAT CAR system. The difficulty of this component was circuitry. The team for this project was inexperienced with wiring and circuits. Many hours must be contributed to understanding circuitry to ensure no components of DAT CAR were damaged.

The motor module was the first component that was successfully implemented in the project. The only complications with the motor module component were the wheel movements when they are activated. Some of the wheels would not turn in the proper direction to initiate movements of the chassis in the respective direction. The solution was just trial and error on the Python code that activated the wheel motors until they function properly.

The camera component was pointed to the front of the DAT CAR system giving the user a view of the front of the system in order to view and traverse the environment. The camera module is located right next to the sensors, this allows for the user to point the camera at the object they would like to gain data from and the sensors are also then aligned. The GUI then displays the values in large text and formatted in a way so that the user can easily understand. Before utilizing the camera\_pi.py import file, Motion was used. Motion is a highly configurable program that monitors video signals from many types of cameras [10]. However, there were many complications with the program. As a result of the dilemma that occurred with Motion, it was found to be more efficient to import the camera\_pi.py file.

The original database was going to an SQL one. However, there were many complications that occurred with the SQL database. As a result, a text file was used to append sensor data. The text file is saved on the Raspberry Pi system and can be accessed from a separate computer using PuTTY. Another method to access the text file is to connect the Raspberry Pi to a monitor and navigate to the directory that the file is located at.

The web server component was troublesome at first. Initially, the project was using Apache, which is a deployable webserver to run Python applications. The difficulty for the project arose when multiple components had to be implemented with the Apache webserver. Upon further research on deployable web server tools, it was found that using Flask lead to a successful establishment of a web server. Flask is a Python web framework built with a small core and easy-to-extend philosophy [11].

The GUI on the client software was also redesigned. For the new GUI design, the display is on the left side of the monitor with the directions on how to use the system on the right as well as the buttons used to collect and save the data of the environment. The team has changed the design so that a user can have an easier time reading the displayed data as well as knowing how to control the system.

There is much potential for expansion in this project. A suggestion is to update the software used for the database. SQL is an example that can replace the text file database. As of now, the movements of DAT CAR is restricted to a linear plane. Extensions to DAT CAR giving it airborne qualities is a highly desirable trait because it is no longer be restricted on ground surfaces. Additionally, DAT CAR does not have defense measures against the other autonomous robots or organisms in various environments. Possible extensions could be adding a flamethrower to the chassis. A flamethrower is a military assault weapon that projects a stream of blazing oil or thickened gasoline against enemy positions [12]. A weapon like a flamethrower would greatly increase DAT CAR’s survival in unknown exploration expeditions where predators may be present.

**6.2 2020 Project**

DAT CAR was constructed through the research and study of software engineering, robotics, and networks. It was found that DAT CAR reflects innovative technologies for the use of safe exploration and the study of environments from a remote location. However, there are many discrepancies that are currently ongoing with the device. The cloud service for DAT CAR is currently being worked on. In its current state, there is functionality with the cloud services, but it is unrefined or is in a state that is not desired. Furthermore, the set up for a user to install the software properly may be complicated for users. There is still maintenance to be done on the project.

A possible suggestion to add on to the project would be a way to move the DAT CAR camera. As of right now, the DAT CAR can spin itself in a circle to see its surroundings, however, if the car was to stay stationary and revolve its camera, then this would allow for the system to acquire data surrounding the car without having to spin the chassis. Adding this would be extremely helpful if DAT CAR was in a potentially dangerous spot. For instance, if the rover was on the edge of a cliff.

**7. References**

[1] Lucy Hattersley, “Raspberry Pi 3B+ Specs and Benchmarks”, The MagPi Magazine, 2018, https://magpi.raspberrypi.org/articles/raspberry-pi-3bplus-specs-benchmarks

[2] Xiaor Geek, “4WD Robot Chassis Kit with 4 TT Motor for Arduino/Raspberry Pi”, Amazon, Amazon, N/A date, <https://amzn.to/2MvFHtV>

[3] Jameco, “Raspberry Pi Pinout Diagram | Circuit Notes”, Jameco, N/A Date <https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html>

[4] Piddler, “L298N Dual H-Bridge”, PiddlerInTheRoot, February 18th, 2017,

<http://www.piddlerintheroot.com/l298n-dual-h-bridge/>

[5] Daniel Dallos, “Android Things Basics: Measure distance with Ultrasonic sensor”, Hackernoon, January 15th, 2017, <https://hackernoon.com/android-things-basics-measure-distance-with-ultrasonic-sensor-3196fe5d7d7c>

[6] Cloud4RPi, “Temperature and Humidity Monitoring Using Raspberry Pi”, Instructables, N/A Date, <https://www.instructables.com/id/Temperature-and-Humidity-Monitoring-Using-Raspberr/>

[7] Don Wilcher, “How to Build a Raspberry Pi Camera”, AllAboutCircuits, 2016, <https://www.allaboutcircuits.com/projects/how-to-build-a-picamera/>

[8] lastminuteengineers, “HOW HC-SR04 Ultrasonic Sensor Works & Interface it with Arduino”, Last Minute Engineers, N/A date, <https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/>

[9] Marcelo Rovai, “Video-Streaming-With-Flask”, Github, January 26th, 2018, <https://github.com/Mjrovai/Video-Streaming-with-Flask/blob/master/camWebServer/camera_pi.py>

[10] Motion, “About Motion”, motion-project, N/A date, <https://motion-project.github.io/index.html>

[11] Matt Makai, “Flask web development, one drop at a time”, Full Stack Python, 2012-2019, <https://www.fullstackpython.com/flask.html>

[12] The Editors of Encyclopaedia Britannica, “Flame thrower”, Britannica, N/A date, <https://www.britannica.com/technology/flame-thrower>