



COEN 315: Web Architecture & Protocols


# PROJECT REPORT – CAMERA ON WHEELS

## “Camera On Wheels”

By Team - RCTeer

Swastika Raghava Bhat  
Onur Burak Yildirim

Santa Clara University, School of Engineering  
25<sup>th</sup> March 2017



## Table of Contents

<b>Abstract .....</b>	<b>3</b>
<b>Introduction .....</b>	<b>4</b>
<b>Background .....</b>	<b>4</b>
<b>Purpose.....</b>	<b>4</b>
<b>Theoretical basis and background .....</b>	<b>5</b>
<b>Research to solve the problem .....</b>	<b>5</b>
<b>Related products .....</b>	<b>5</b>
<b>Approach .....</b>	<b>6</b>
<b>Our Solution.....</b>	<b>6</b>
<b>Video Streaming.....</b>	<b>7</b>
<b>Maneuvering the car .....</b>	<b>7</b>
<b>Hardware Setup .....</b>	<b>8</b>
<b>Source Code .....</b>	<b>11</b>
<b>Firmware.....</b>	<b>11</b>
<b>Restful web server .....</b>	<b>12</b>
<b>Web and mobile web applications.....</b>	<b>13</b>
<b>Conclusions .....</b>	<b>13</b>
<b>Findings.....</b>	<b>13</b>
<b>Future Work.....</b>	<b>13</b>

## Table of Figures

<i>Figure 1: Basic flow diagram .....</i>	<i>6</i>
<i>Figure 2: Arducam connected to ESP8266 .....</i>	<i>8</i>
<i>Figure 3: Pin Configuration for Motor Shield .....</i>	<i>9</i>
<i>Figure 4: Web Application screenshot – System is Off .....</i>	<i>10</i>
<i>Figure 5: Web Application screenshot – System is On .....</i>	<i>10</i>
<i>Figure 6: Mobile Web App screenshot - System is off .....</i>	<i>11</i>
<i>Figure 7: Mobile Web App screenshot - System is On.....</i>	<i>11</i>
<i>Figure 8: : Block diagram representing technologies used .....</i>	<i>12</i>

## Table of Tables

<i>Table 1: Pin Configuration for connecting ESP8266 to Arducam OV2640 .....</i>	<i>8</i>
----------------------------------------------------------------------------------	----------

## Abstract

This is a technical document describing in great detail the implementation of the “Camera on Wheels” project. This is an IOT project involving a remotely controlled car attached with a camera. A dc motor toy car is connected to an esp8266. The car also has an Arducam attached to it. The Camera transfers a video stream to the user showing the car’s surrounding. A web application and a mobile web application have been developed to help the user control the car. The continuous video stream helps the user to control the car without being in its visibility range. The car motor and the camera act as a client and are connected to a webserver in the cloud. This architecture allows the car to be controlled remotely from anyplace in the world with internet connectivity. This project is a proof of concept that an esp8266 can be used as a client to transfer image data and to receive motor commands with minimum latency. The applications of this implementation are many and have been discussed briefly in the introduction section.

## Introduction

Suppose that you are in your office and want to see how your pet dog is doing all alone at home. You can install a pet camera to watch your dog. But those cameras are very expensive. You might install one in the main room but your dog can be in any room of the house. Or imagine a situation where you and your friend live some distance apart and cannot meet to race toy cars. Or maybe you are curious to know what is there inside a rat hole. This project has found the answer to all the above problems. With the help of this project document you can build your own 'camera on wheels' implementation of a remotely controlled car with a camera.

There are a lot of wireless toy cars in the market. But they have two main disadvantages. First, the user needs to be in the visibility range of these cars to efficiently maneuver them. Secondly, the user needs to be in the restricted network range of these cars. This project overcomes both of these obstacles. We augmented a remote controlled car with a wireless camera. The camera streams video of the environment in front of the car. This stream is displayed on the user's device. Using this feed the user can decide on the future direction of the car. We programmed the camera on wheels to connect to a remote web server hosted on aws, to upload the video stream and access driving control commands. A user can also connect to the same server, via a web-browser or a mobile app, to access and control the camera on wheels. Unlike the pre-existing solutions, this client-server architecture expands the network range of the car. In other words, the user can control the car from anywhere as long as both of them are connected to the internet.

We have built a web application to enable the user access to the car from his laptop or desktop. The mobile phone application allows the user to control the car from the comforts of his couch. The user can also use their phone's accelerometer to maneuver the car forward, backward or sideways. Both these user interfaces have buttons (start system, start engine, stop engine, forward, backward, left and right) for controlling the car and a panel to view the live video stream.

## Background

### Purpose

The main purpose of this project is to develop a prototype of a remotely controlled car using an ESP8266 Wi-Fi breakout board, based on a client - server architecture. This project encompasses all four components that we have learnt as part of this course. It has a front end web and mobile app interface for the user to view the environment around the car as well as control its speed and direction. The web backend consists of the server hosted on the cloud. It acts as the facilitator of information flow between the car and the user. The car itself has a camera and a motor controlled by ESP8266. A real time information is transmitted to the user over the internet.

## Theoretical basis and background

Generally, on the web, “Push” technology is used. Here the client sends a request to the server and the server responds back with the required data. The request can be for a text file, image, etc. When the client sends a request, a connection is opened. The server responds to the request and the connection is closed. In most of the cases, a simple HTTP request method “GET” is used for retrieving data from the server. On the other hand, to send data to the server, the HTTP request method “POST” is used. All this works fine when the communication between the server and the client is limited.

But in scenarios where we need to continuously send data, the overhead of opening and closing the connection introduces latency. To overcome this the websocket technology was introduced as part of the HTML5 specification. Websockets allows for a 2-way real time communication between the client and the server. Here after the connection (handshake) has been established, the communication channel between the two remains open until it is interrupted or disconnected.

In our project, we need to continuously send the image data from the ESP8266 acting as a client to the server and in real time pass this same data from the server to the web browser (client). We have thus made use of the web sockets API to achieve this real time communication with minimum latency.

## Research to solve the problem

In all the projects available online, the car or the camera both are programmed as web servers. In contrast, we have implemented the car motor and the camera as clients transferring and receiving data from a cloud based server. There are a few other projects in which the camera acts as a client but in that case it captures images every few seconds and stores it in memory for later use. None of the projects transmit the image data at real time like us using the ESP and the arducam. Also, there are no libraries available that controls the car motor using this architecture. However, the libraries used in these products did provide us with a starting point. Few of the related products are described below.

## Related products

- **Arduino Camera Module:** [5] This instructable includes information on how to attach ArduCAM-Mini to Arduino-Nano and transfer pictures to a PC. It also explains how to attach ArduCAM-Mini to ESP8266 Wi-Fi module, use it as a webserver and display pictures on a web browser. In this project they have

programmed the ESP8266 to upload pictures to a webserver using http request method POST.

- **Remote Car:** [6] This project includes information on controlling a car remotely. However, here the ESP8266 acts as a webserver and thus it does not adhere to our architecture. However, it provided good information on how the direction and speed of the car can be controlled.

## Approach

### Our Solution

Figure 1.1. Is a block diagram representing the interconnection between all the main components.

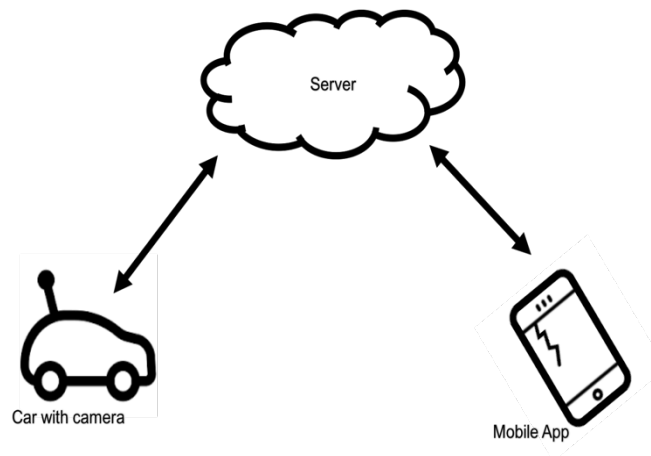


Figure 1: Basic flow diagram

As shown in the figure1, we have three basic components. The car with a camera consisting of all the hardware ('thing' component of the IOT), the cloud based server, and web/mobile app for user interface. The hardware component can be further divided into the two main components, the video and the motor.

First, the DC motors of the car that control the direction and speed of the car. This motor is connected to a motor shield which is in turn connected to the ESP8266 with Wi-Fi module. Second is the camera on top of the car that continuously captures and sends the video feed. This camera is a two megapixel Arducam OV2640 connected to another ESP8266 with Wi-Fi module. The ESP8266 connected to the Arducam acts as a client. It requests a websocket connection to the web server. For the ease of understanding the functionality, we can consider the two activities "streaming video" and "maneuvering the car" separately.

## Video Streaming

Let's consider this function independently. It consists of the camera hardware, server and the user interface. The ESP8266 connected to the Arducam (camera hardware) sends a request to the server to initiate a websocket connection. The server accepts the request and after the initial handshake, the websocket connection is established. The Arducam then starts capturing images. Arducam converts the raw data into an image in jpeg format. As soon as an image is captured, it is buffered and then packaged in buffers of predefined size. Through trial and error, we found that a buffer size of 4096 along with a resolution of 160X120 gives optimum video clarity. The buffered bytes are then converted into binary data and sent to the web server. Next, another image is captured and sent. This is performed in a loop continuously to mimic video streaming. To differentiate between each image, we send a text message marking the end of one image data.

In the user Interface end, there is a button called "Start System". On clicking this button, a request for a websocket connection is made by the browser/client to the server. The server accepts the request and after initial handshake a websocket connection is established. Now the server passes the whole image data received from the camera to the browser. The server will keep sending the image data one after the other mimicking a video. The user views this video and can control the car accordingly.

## Maneuvering the car

Let's consider this function independently in its entirety. Similar to video streaming, it consists of the motor hardware, server and the user interface. As mentioned earlier, a websocket connection is established between the browser and the server. The UI also has a "Accelerometer Toggle" button. Using this button, we can toggle between the direction buttons on the web application and the phone's accelerometer. As tilting the phone for controlling the car is more intuitive, we added this feature. It behaves in the same manner as that of the forward, backward, left and right buttons on the UI. On clicking on any of these buttons, the client side JavaScript converts the command into JSON object and passes the same to the server.

In the car the motor shield is connected to the ESP8266 with Wi-Fi module. The ESP8266 connects with the server using RESTful API service and keeps on sending "GET" request to the server in a loop. When the server receives the JSON object it passes it to this ESP. The ESP then in turn controls the motor accordingly.

## Hardware Setup

We have used the following hardware components:

- Two ESP8266 with Wi-Fi module
- Arducam OV2640 2 MP
- Power Bank
- Motor shield
- USB chord
- Connecting wires
- Standard half sized breadboard

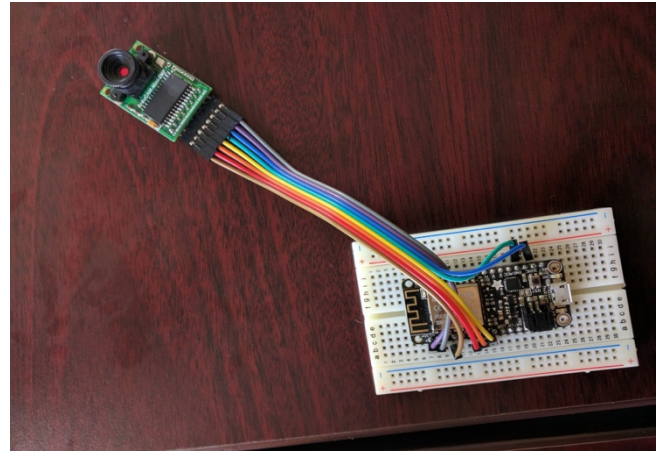


Figure 2: Arducam connected to ESP8266

Table 1: Pin Configuration for connecting ESP8266 to Arducam OV2640

HUZZAH Feather ESP8266	Arducam Ov2640
#16	CS
#13	MOSI
#12	MISO
#14	SCK
GND	GND
3V	VCC
#4	SDA
#5	SCL

We have used the breadboard to connect the ESP8266 to the Arducam. We have also used a readymade car kit containing a chassis, 2 DC motors and 3 tires to form the car framework [4]. A second ESP8266 is connected to a motor shield which is in turn connected to the motors of the car. The 2 ESP's are supplied power by a power bank. We used an easily available power bank used for charging phones.

The arducam is connected to the ESP8266 as per the pin configuration given in table 1. We used a breakout board to make these connections. Remember, the pin number and pin position of the ESP8266 will differ for each brand. The given configurations are for Adafruit HUZZAH Feather ESP8266. Figure 2 shows the actual HUZZAH Feather ESP8266 connected to Arducam OV2640. Figure 3 shows the connections for a motor shield. There are no positive and negative points for the DC motor. But reversing the power supply will cause the motor to rotate in the opposite direction.



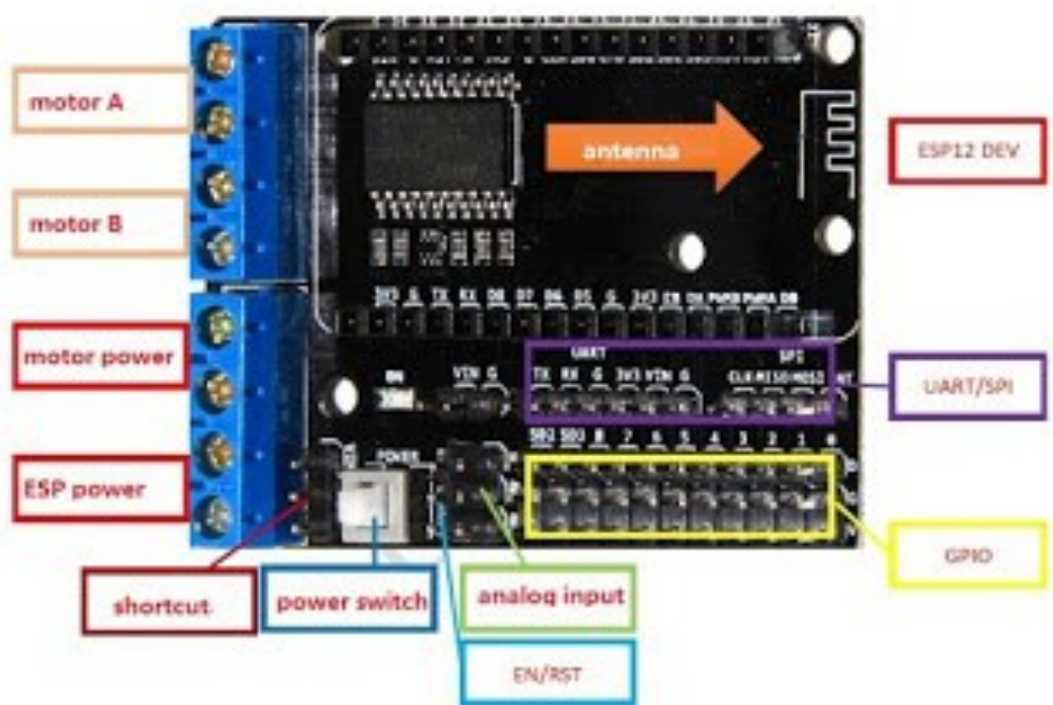


Figure 3: Pin Configuration for Motor Shield

## Results

The Screenshot of the web application can be seen in the figure 4. As shown the “Start System” button is used to connect to the server. On clicking that button the video will start streaming as can be seen in the figure 5. When “Accelerometer Toggle” is enabled the phone’s accelerometer is used. When it is disabled, the direction buttons on the screen can be used for controlling the car. Figure 6 and Figure 7 show the screenshot of the mobile web application.



Figure 4: Web Application screenshot – System is Off

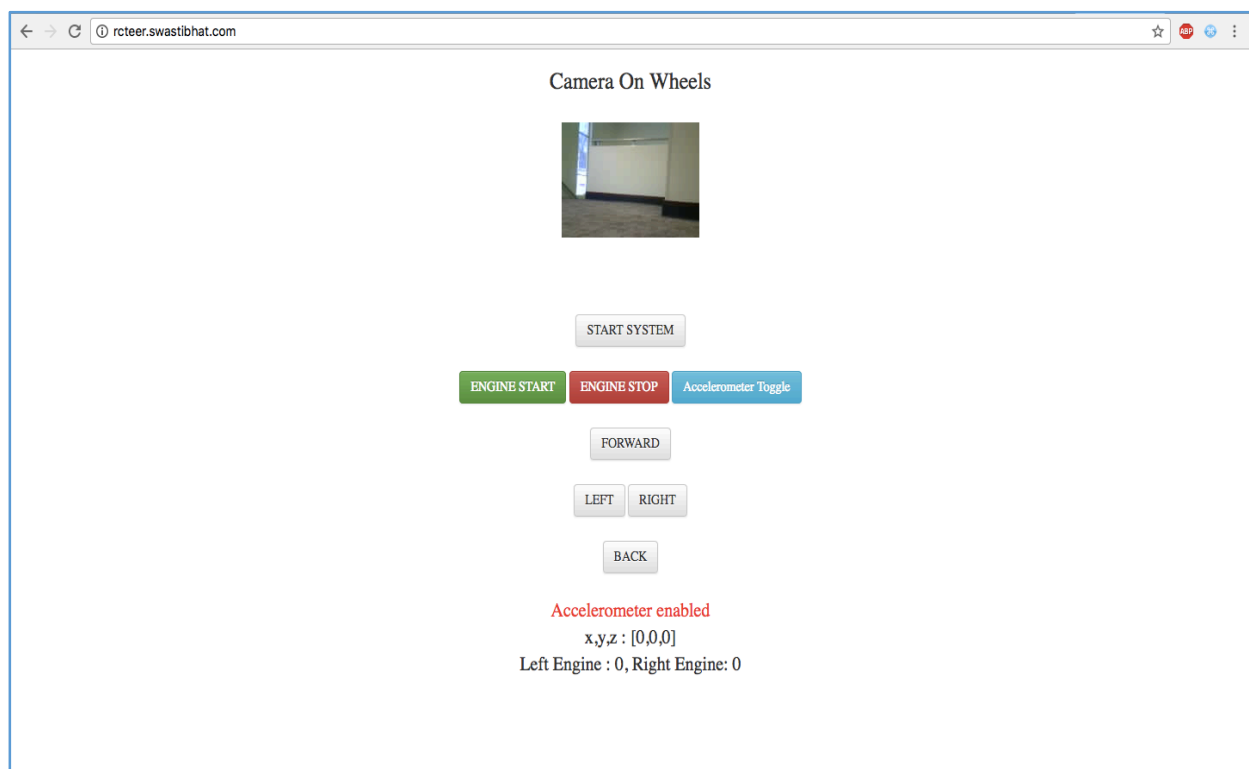


Figure 5: Web Application screenshot – System is On

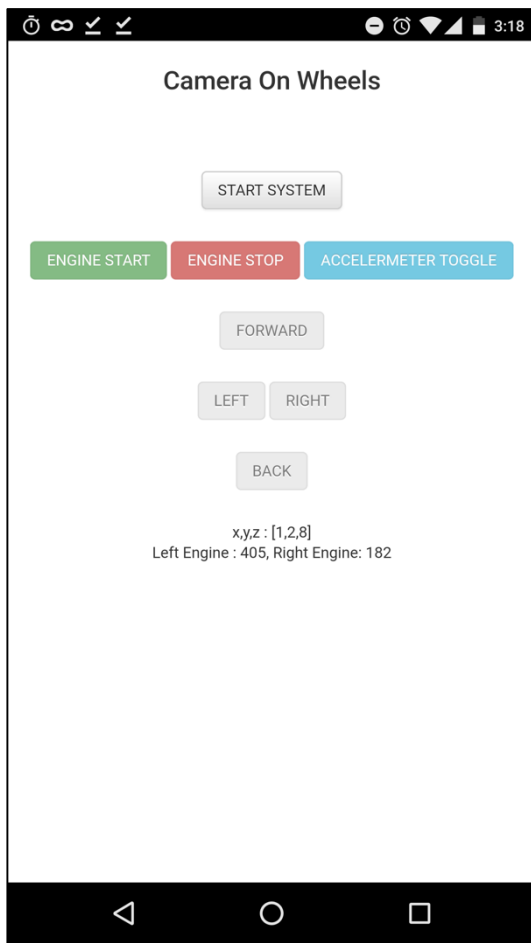


Figure 6: Mobile Web App screenshot - System is off

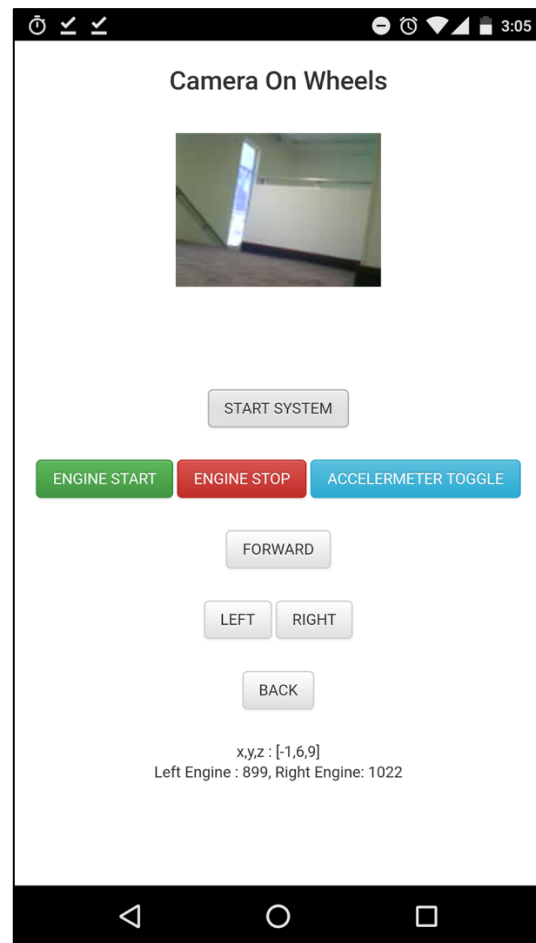


Figure 7: Mobile Web App screenshot - System is On

## Source Code

### Firmware

The two ESP8266 have been programmed in C++ using Arduino IDE v1.8.1. The commonly used library for arducam can be found in GitHub [1]. We used the "ArduCam\_ESP8266\_OV2640\_Capture" example code for capturing an image and converting to JPEG file format. WiFiClient from Arduino's ESP8266WiFi library was referred for connecting the ESP8266 to the specified Wi-Fi connection.

Next, to test a websocket connection the most commonly used example "WebSocketClient\_Demo" from the ESP8266\_Websocket library [2] was used. However, it sends data in the raw format and hence wasn't very useful. The node js websocket in the server side accepts data only in text and binary format. Thus, we had to search for another library that converts the data into binary format before sending it to the server. The example file "WebSocketClient" from the WebSockets [3] library was referred.

However, this library does not work as it is. So we had to build on top of the previously mentioned libraries and used this library to just convert the image data into binary data.

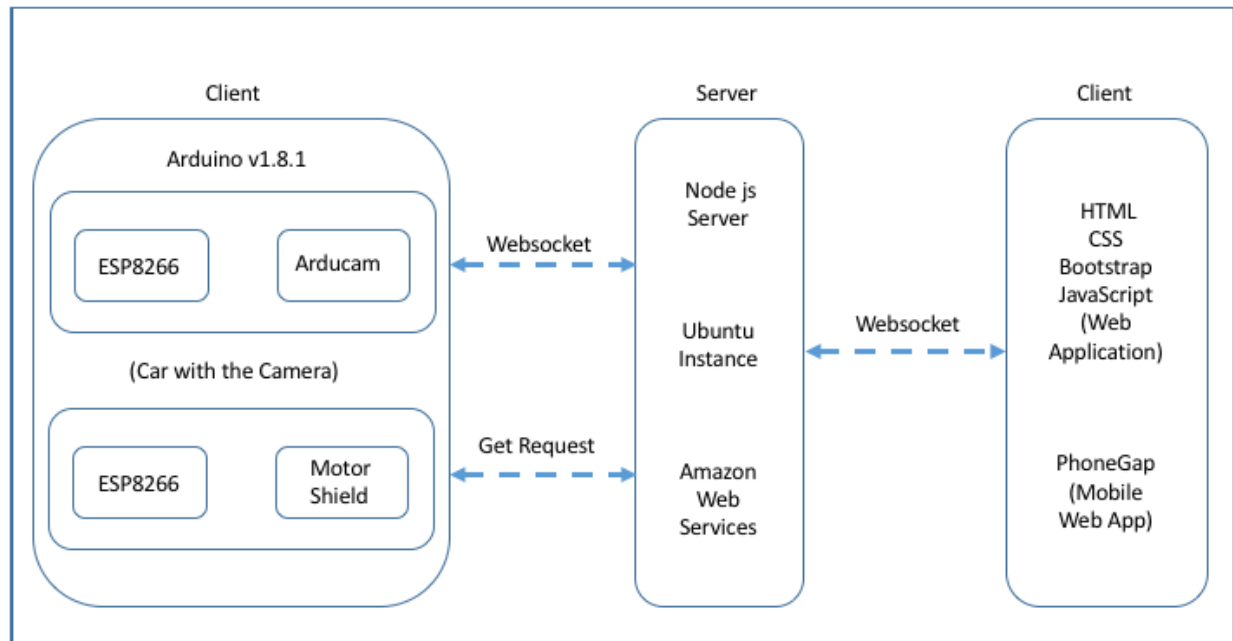


Figure 8: : Block diagram representing technologies used

For the engine side, we understood that we have to send two values to the two engines to set the direction and the speed of the car. In the browser side, we tried different methods to give user better experience and finally we decided on using the accelerometer of the phone. It gives user a better feeling of having control on the car and it is easy to get horizontal and vertical values from the browser's accelerometer API. Moreover, we added some direction buttons with predefined values for users who are trying to control the car from desktop or any phone without accelerometer.

For the ESP part, the kit that we used offers some examples for how to implement a connection between the ESP and the engine. However, all of the examples only contain information for p2p connection between the ESP and the user. We have created an endpoint for ESP to request the current engine value in its loop and on the server side we updated the engine value with the last value that is received from browser (client).

## RESTful web server

We created a subdomain named `rcteer.swastibhat.com`. This subdomain has been linked to an Ubuntu instance created in cloud using amazon web services. An apache server was installed in this instance and was configured to redirect the requests to the node js server.

As stated in the Node.js website, Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-

driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. For these reasons we used Node.js for creating the web server.

In the Node.js server we have used the “websocket” module to establish a websocket connection with the client. In addition, we have used “express” to establish a connection between the ESP8266 connected to the car’s motors and the web server.

## Web and mobile web applications

A combination of HTML, CSS, Bootstrap, JavaScript and JQuery library have been used for creating the user interface for the web application. The User interface consists of a display panel for the video to be displayed. It also has a control panel that houses a number of buttons for accessing the car. The PhoneGap application was used for converting the web application into a native android application. Figure 8 provides an overview of the various technologies used in this project.

## Conclusions

### Findings

This project showed us that it is possible to program ESP8266 as a client and transfer data over a websocket. It increased our knowledge of the way a websocket works. We learnt how to handle content type “multipart” data. We overcame the lack of documentation problem through intense research and experimentation. These obstacles that we faced provided us with important lesson in truly understanding the “internet” as well as the “things” component of the IOT infrastructure. All the related work have been uploaded to GitHub[14].

### Future Work

Though there are numerous directions in which this project can be continued, we have identified three main things. They are

- **Fish eye lens for the camera:** We need to find if any fish eye lens camera that are compatible with the ESP8266 are available. This will help us in getting a better view of the car’s environment which will in turn help the user maneuver the car properly while controlling it remotely.
- **Actuator to control camera direction:** This will increase the usability of the car as a 360-degree view of the car’s surrounding can be viewed from the user’s device.
- **A fool proof car casing, better frame -** The car frame that we currently used in the project is not very stable. It has three wheels. The front wheel rotates in 360 degrees. This sometimes misdirects the car. A better frame will help increase the car’s performance. Also a good casing will protect the hardware inside the car against tampering.

## Reference

1. <https://github.com/ArduCAM/Arduino/tree/master/ArduCAM/examples/ESP8266>
2. [https://github.com/morrissinger/ESP8266-Websocket/tree/master/examples/WebSocketClient\\_Demo](https://github.com/morrissinger/ESP8266-Websocket/tree/master/examples/WebSocketClient_Demo)
3. <https://github.com/Links2004/arduinoWebSockets>
4. [https://www.amazon.com/INSMA-Chassis-Encoder-Battery-Arduino/dp/B01BXPETQG/ref=sr\\_1\\_1?s=toys-and-games&ie=UTF8&qid=1490458172&sr=1-1&keywords=remote++car+kit+Insma](https://www.amazon.com/INSMA-Chassis-Encoder-Battery-Arduino/dp/B01BXPETQG/ref=sr_1_1?s=toys-and-games&ie=UTF8&qid=1490458172&sr=1-1&keywords=remote++car+kit+Insma)
5. <http://www.instructables.com/id/Arducam-Mini-With-ESP8266-Wi-Fi-Is-Amazing/>
6. <https://blog.squix.org/2015/09/esp8266-nodemcu-motor-shield-review.html>
7. <https://github.com/onurburak9/esp8266-engine-websocket>
8. <https://github.com/onurburak9/esp8266-arducam-nodejs>
9. <http://www.instructables.com/id/Arducam-Mini-With-ESP8266-Wi-Fi-Is-Amazing/>
10. <http://www.arducam.com/arducam-supports-esp8266-arduino-board-wifi-websocket-camera-demo/>
11. <http://cjhrrig.com/blog/creating-your-own-node-js-websocket-echo-server/>
12. <http://www.instructables.com/id/Motorize-IoT-With-ESP8266/?ALLSTEPS>
13. <https://github.com/Links2004/arduinoWebSockets>
14. <https://github.com/onurburak9/esp8266-arducam-nodejs>