**Spring 2016**

**CS 250: Software Engineering**

**CAR**

**Group Report**

**Project Description/ Overview**

The main idea was to build a remote car. A toy RC card that can be controlled through a simple web application. The focus was to build a completely wireless car that was independent of any external connections including power. We wanted to build a car that looked visually appealing and had efficient software and hardware integration. Some of our stretch goals included a well-designed interface, live stream and other sensor integration such as infrared.

**Hardware requirements**

* Lightweight enough to move at reasonable speeds while powered by a 9V battery.
* Can be controlled wirelessly.
* Self-contained; all components must fit within car’s case.

**Software requirements**

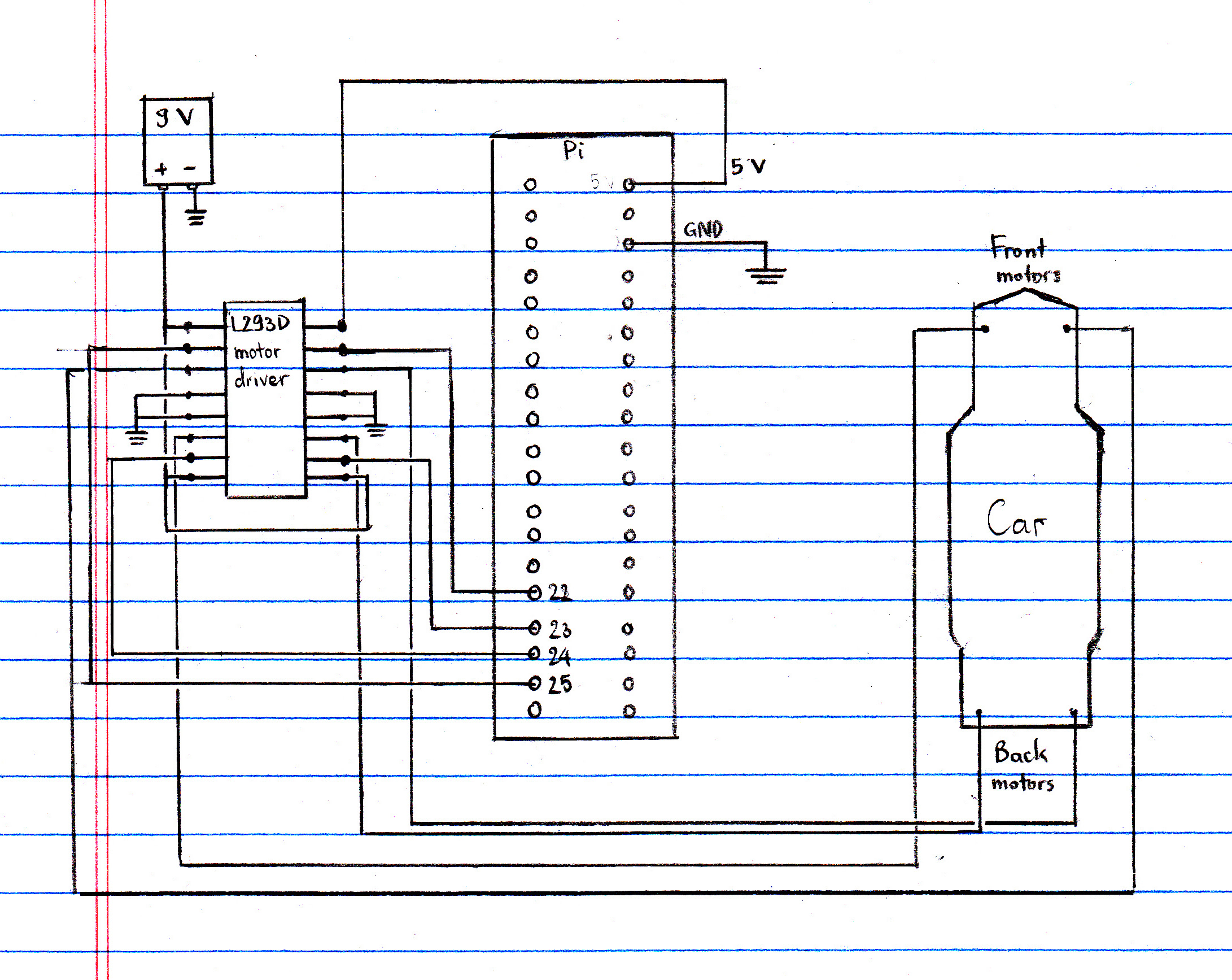
* Host a web app that controls car’s motors and provides a live stream from car’s camera.
* The web server must draw a minimal amount of power so that the Raspberry Pi remains functional while on battery.
* Live stream must have minimal delay to facilitate easy navigation.
* Must be easily accessible over wireless LAN.

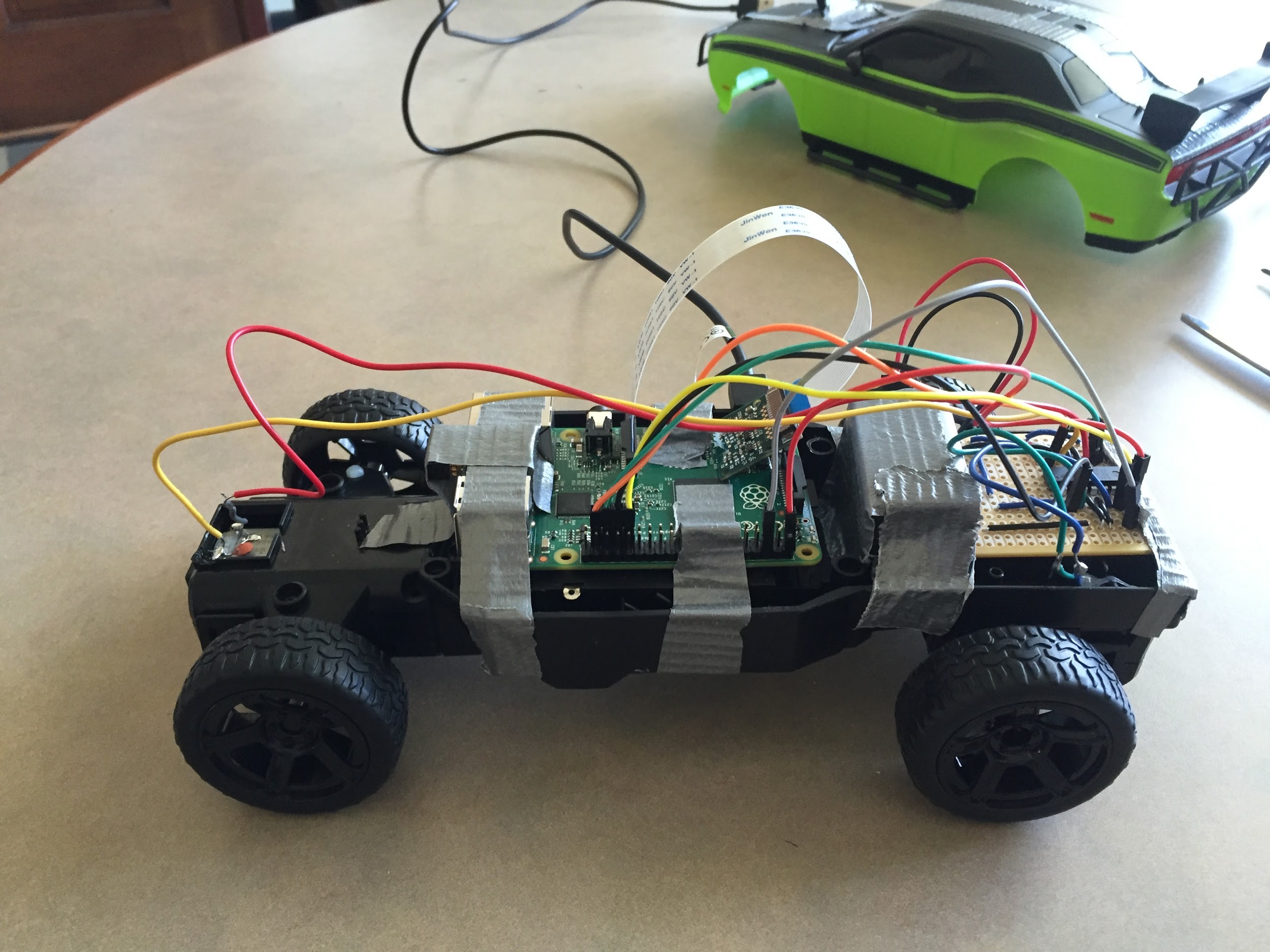
**Design Documentation**

Website:

* Minimalistic design
* Simple interaction with car controls
* Display key information about project

Diagram below shows the layout of between the H-bridge motor driver, the car and the Raspberry Pi 2



Picture below showing final assembly of all components. (Motor control cables soldered onto custom circuit board).

**Group Reflection**

We had some earlier experience working with these types of components. Our first step was figuring out the parts that we needed to purchase: batteries, RC car, wires, solder, bread boards, circuit boards and a Raspberry Pi 2. As per the project requirements we wanted to build something completely independent and self-contained. The Pi is a highly versatile and affordable mini computer that is lightweight and has ample computational power to host a website that is accessible over wireless LAN.

The project itself was interesting for both of us as we are both fascinated with remote systems such as driverless cars and especially drones. The beginning of the semester was spent on understanding the car and infrastructure that needed to be present to allow control of car from a web app. After setting up the Raspberry Pi, we build our first circuit over spring break on a breadboard for prototyping. However upon soldering the circuit together and with the car, the motors were not getting enough power from the car’s built-in battery pack and that pushed us back to the drawing board. In our second attempt we switched to using a separate 9V battery. Soldering however was by far the most challenging and time consuming task. We had limited equipment present and the work itself is very delicate and leaves very little room for error. Abhijit proved to have very steady hands for this task. Once we had the soldering done, everything really fell into place quite fast.

We both worked together on the project and often helped each other with tasks. The hardware knowledge was shared with Abhijit and the both of us contributed in building the physical circuit. Tom helped to write the control script in the PHP that changes the state of the GPIO pins on the PI and Abhijit helped to build the front-end and the javascript components. We both worked together all of the times in the same room since we both required each other’s assistance in making decisions.

The software component of the project was quite interesting. The part both of us found challenging was making decisions based on functionally, design and especially performance. Though Pi is a powerful machine for its size, our challenge was to stay as light as possible to ensure that the ultimate experience for controlling the car was seamless. We set up the Raspberry Pi to host a website with Apache. In order to control the Pi’s General Purpose I/O (GPIO) pins, we installed WiringPi, a C library which includes a CLI utility to interact with the GPIO pins without sudo access. Tom wrote a PHP script to process GET requests and interact with the aforementioned utility to send commands to the motor driver. He also wrote some Javascript for the web app to send appropriate GET requests to the PHP script when a user controls the car via arrow or WASD keys.

One of the biggest problems we ran into was getting the car onto ClarkWiFi. We wanted to ensure that the website hosted on the Raspberry Pi was accessible via a friendly domain name. While we were working on the car at home, we configured the Raspberry Pi to automatically broadcast its IP address on a multicast DNS. This approach unfortunately did not work while the Raspberry Pi was on ClarkWiFi. To deal with this issue, we had to set up a hostname through a free dynamic DNS service (No-IP) and install their utility onto the Pi, which updates the hostname to point to the Pi’s IP address every minute.

**Individual Reports**

**Abhijit’s Contribution**

* Helped in wiring and soldering the Raspberry Pi
* Researched Raspberry Pi and helped plan wiring of Pi, motor driver and car motors.
* Researched on different integration of sensors with Raspberry Pi
* Set up the Raspberry Pi and Apache Server
* Configured the WiringPi library and the Pi’s Camera to allow broadcasting of live stream
* Wrote front-end components of website and Javascript
* Helped to configure the Pi to connect with ClarkWifi

**Individual Reflection**

The overall experience in making this remote car was extremely fun. The software we have built is quite adaptable with little change and can be used with other devices with similar controls. There biggest shortfall of this project is unfortunately the car itself. The car is of very poor build quality and has a very large turn radius and that is due to a really weak front motor installed by the manufacturer.

The legendary boxer Mike Tyson said, “Everybody has a plan until they get punched in the face”. I think software engineering is quite similar. We all start out with a plan and spend significant time planning until we start the actual work required for the project and realize that our plan was “not enough”. I think modern software engineering is a very and needs be a wholesome process where user, functionality and what is being built should be reviewed frequently. I believe once you actually start building and working towards set goal is when you realize the difficulties. Having foresight that predicts all problems and includes them in the requirements in near impossible as something will always break.

In this project Tom and I both followed a very agile approach where we worked on certain parts to synthesize what we needed to achieve our goal. We both worked together since it was just the two of us in the group. However I think since our group was small and the project had a very limited scope it allowed us to be iterative with our approach in finding and trying different libraries and methods to achieve our goal. Once we found the resources and the knowledge we needed, it was relativity easy to translate it into a working product. Overall it was a positive working experience and I can definitely see the benefits of creating artifacts like requirements (design, hardware) and doing the SRS process for large projects with many team members and many different requirements.