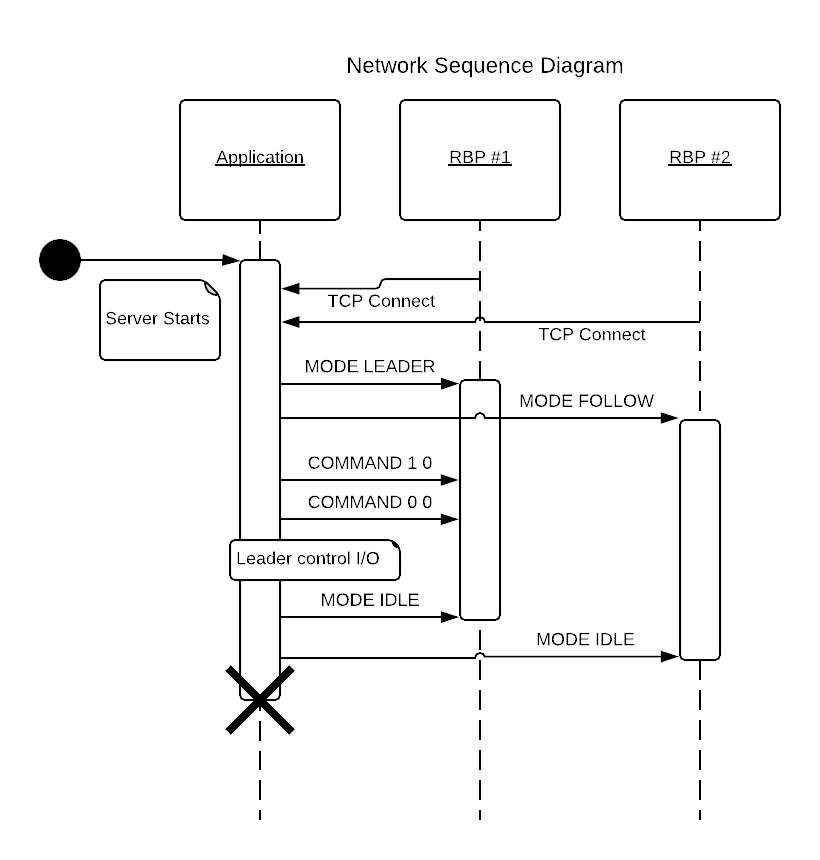
The general network design for the project will be a desktop application that acts as a server, communicating with two Raspberry Pis running the vehicles over LAN. The general communication sequence was decided for the project: Where the RBPis connect to the desktop application server, which then sends messages to the RBPis.



The base SunFounder Pi-CarV project uses a Django server in order to send data over HTTP. Although research was done into this project, we decided it was too heavy for our purposes, and none of us have any experience with Django or with sending data over HTTP. Thus, we began looking for other solutions.

Initial research for network implementation was done with straightforward socket programming. We were only planning to send small amounts of data, and by sending data of our own format it’d be simple to communicate between the Raspberry Pis running Python and the desktop application running C#.

Python’s official documentation is being studied for reference with socket programming:

<https://docs.python.org/3.7/howto/sockets.html>

However, this sort of programming is synchronous: Waiting for a response will cause the program to block until it receives a message. In order to have a follower vehicle listen to commands while still controlling the vehicle, research was done into asynchronous solutions.

Python has an asyncio module that’s used for this sort of asynchronous IO. This documentation was referenced for future implementation:

<https://docs.python.org/3/library/asyncio-stream.html#asyncio-streams>

We had then started creating a message protocol to send over the packets

The message protocol for packets sent from the server to the Raspberry Pi.

There are two forms of messages: mode messages which set the operating mode of the Raspberry Pi, and com messages which are input for the leader Raspberry Pi’s motion.

Messages are sent as strings encoded in ASCII. Their arguments are delimited by spaces and interpreted by the Raspberry Pi clients.

Mode messages:

MODE IDLE

The RBPi will simply idle until it is changed to a new mode.

MODE LEADER

The RBPi will listen for COMMAND messages to determine its movement

MODE FOLLOW

The RBPi will use it’s camera to follow the leader RBPi.

Command messages:

COMMAND 0 0

The two floating point numbers proceeding COMMAND control the throttle and the steering respectively. They are clamped between 1 and -1. For the throttle, 1 is max forward speed and -1 is max reverse speed. For the steering, -1 is far left and 1 is far right