Team 18 Background Research

Andrew Bryant, Danny Dutton, Benjamin Singleton, John Spataro

September 18, 2016

1 Network Communication

For two devices to communicate over WiFi, a application layer protocol is needed to run on top of the already existing TCP/IP layer provided by the WiFly and WiPi peripherals. For the two rovers to talk, they need to pass messages containing a chunk of data along with a description of the data.

There are many types of existing protocols that can perform the type of message passing needed but they all have caveats. Passing simple JSON messages inside of TCP packets is one of the most intuitive methods. However, this means there needs to be handlers running on all the devices that can decode the data into the JSON format as well as decode the format down into the parts the device actually cares about. Google's Protocol Buffer is similar to JSON but allows for more customization. Again, this would need additional resources to encode and decode the data.

Due to how simple the messages sent between rovers and devices will most likely be, having a simple TCP packet containing a string of a yet to be determined length. In this string, the first chunk of bits can be used as an initial description of what's being sent (e.g. position of a rover) or can be used as a basic instruction (e.g. turn left). The number of bits used for this chunk determines how many types of instructions or descriptions are possible.

Following this first chunk of bits is the data section. It contains whatever data is to be passed to the device, described by the first chunk. If more instructions or descriptions are needed, a type of instruction would be used that allows for a second description section that follows the first.

In effect, this works similar to machine code, the instructions and literals are encoded in a single message. Due to the nature of TCP packets, the sender address can be determined such that the name of the device sending the packet doesn't need to be encoded in it, freeing space for more data.

Additionally, these messages can be sniffed on the network using a tool like Wireshark. The tool can be supplied with the schema for the message such that they can be easily be viewed for debugging as they are being transmitted.

2 Line Following

The SparkFun Line Following Array (LFA) is a device designed to provide information on the location of a line within the array. The sensor works by relaying information about the eight sensor "eyes", specifically which eye is currently sensing a contrasted line.

The LFA is attached to the front of the rover. Its maximum detectable line width is 3/4 of an inch. The power source for the LFA is 5V at a maximum of 185mA.

The protocol for the LFA is I2C. Example code is given in a Github repository provided with the LFA. Features of the LFA include a physically adjustable sensitivity potentiometer and toggleable sensors.

Sunlight and shadowing can cause issues with LFA devices. Shadowing presents a false-contrast condition, while sunlight can overwhelm the sensor array with UV light, causing sporadic readings.

In fact, it is useful to low-pass filter the results of the sensor in software to prevent such issues and tightly control the lighting environment.

Another issue with LFA devices is the case where no line is detected at all. In this case, information about the last known line position needs to be taken into account, or the LFA will cause a rover to lose its location. This is usually done by keeping a record of some previous sensor states, and when the "no detection" state occurs the record must be accessed to determine the best route to relocation.

3 Path Finding

One of the key requirements for the ghost rover's functionality is its ability to quickly and accurately determine the best path through the playing field. There are a number of challenges involved in this process, including the limited memory of the board, the difficulty in acquiring the current location of the player rover, and the need to make good guesses of how a non-deterministic agent (i.e., the player) would act. These challenges help to narrow down the innumerable path-finding solutions out there. Taken together, it is clear that the path-finding algorithm used must be both quick and not cost much memory, as it will have to be readjusted continuously throughout its operation.

The simplest, and most naive solution, would be a backtracking solution that begins with the current position, and then tries possible paths to the target position in a recursive fashion. This type of search "backtracks" whenever it fails, going back to the last intersection that was considered. This is a very computationally-intensive search that is good for exhaustively finding all possible solutions to a problem, but it is not an efficient algorithm for finding the optimal solution.

A much faster algorithm for this type of application is the A* search. This algorithm considers paths based both on how much closer they get the rover to its destination and how much it would have to travel to achieve this, eliminating the need for considering less efficient paths. The efficiency of A* is well-documented, and its lower rate of branching would also help to reduce the memory footprint.

The downside of the A* search compared to the backtracking algorithm is that A* requires an ability to estimate the "cost" of a particular route. Whether this is easy or difficult depends on the representation of the playing field. Whether the playing field's layout is preprogrammed or "learned" through operation clearly has a big impact on the performance of these algorithms. The backtracking algorithm would be a better choice for a learned environment, as A* requires knowledge of the environment in advance.

4 Pixycam

The Pixycam is a small camera that allows you to interpret your surroundings. It can differentiate colors, and can be programmed to look for specific shapes and color cues, as well as calculating angles. It was developed by Carnegie Mellon University and Charmed Labs, and can interpret its surroundings using much less processing power than interpreting a normal camera's input.

The Pixycam would be attached to the ceiling above the playing field. It will be used to tell the location and direction of the rovers. This will allow the 'ghost' rover to make intelligent decisions about how to best chase the 'Pacman' rover.

The Pixycam can be used with a variety of communication protocols, including I2C and UART. The Pixycam comes with software that can already interpret things such as balls or blocks, but

is also programmable to look for user defined objects. It is compatible with a variety of devices, including most microprocessors.

There are some potential problems and limitations to the Pixycam that must be accounted for. It provides fifty frames per second, so if the Pixycam misreads something one frame, code needs to be used to prevent the anomalies from misdirecting the 'ghost' rover. Also, since the Pixycam will be used from a relatively moderate distance, there must be measures taken to tell the true distance an object is using mostly basic trigonometry.