The original challenge was to control an Arduino rover through an infrared remote controller and set boundaries using the ultrasonic sensor and the line follow sensor. Overtime it changed to include more uses in the remote controller like controlling the motor speed.

First thing I need to do before starting to code is learn how to use the infrared remote controller. Each button on the controller emits a unique pulse that repeats and can be decoded by the Arduino. The code is organized by making a switch case that takes in all possible buttons that could be pressed. The IR receiver is on port 12.

I set up the buttons of the IR receiver with up going forward, right turning right, left turning left, and back going back. I then made the asterisk a method of overriding the wall stop and the hashtag a method of overriding line stop.

#define UP 16736925

#define DOWN 16754775

#define RIGHT 16761405

#define LEFT 16720605

#define ANTI\_CRASH 16728765

#define ANTI\_LINE 16732845

I then brought over the necessary methods from previous projects to make the car move like forward(). I used a switch statement to assign each button a certain state.

After testing it, the buttons worked, but only when you pressed them and not when you held them. Also, once the rover got close to a wall it would remain in “close to wall mode” even if the object in front of it was removed. This was because I never made a case to set it back to false.

It feels like most of my button presses aren’t being registered. I found that I had an extra delay call at the start which caused many of my button presses to be ignored. After printing what the IR receiver read, I found that holding any button results in the same number: 4294967295. Since all of the buttons result in the same output when held, I decided not to do anything specific with it.

Since everything was working well, I decided to create a speed setting for each of the numbers. I initially mapped values from 1-10 to 0-255 for the car speed, but I found that less than half of the buttons sufficiently provided enough power to overcome static friction.

I still wanted to have a variety of speeds, but still have all the settings overcome static force. To do this, I used a map function that assigned values 1-10 to a certain value in the range 100-255. I found that the lowest setting worked and still moved slowly.

There are still some problems with the line stopping mechanism. I’ve tried having an override method controlled by the remote, but it was difficult to see if the override was actually on or off. If I had more time I would’ve added or changed how the line sensing worked and when it would actually stop.

Overall I think my use of time was efficient as I accomplished my goal and added on to my original goal. Furthermore, I spent more time writing my initial code which decreased the amount of debugging I needed to do.