RIDWAN HOSSAIN
100747897

2022/05/27

- **Note:** The infotainment system is connected to the PC via the blue usb cable.
- The usb device information was found about the infotainment system using the '**lsusb**' command: '**Bus 001 Device 011: ID 0e8d:201c MediaTek Inc. FF-5000**'
- Connected head unit to campus wifi using student login
- Can enable location spoofing on the head unit as instructed by this source: https://www.howtogeek.com/795277/how-to-spoof-your-location-on-android/
- Before proceeding further, looked into how to obtain location data from the Carla simulator
- **IMPORTANT NOTE:**

  It was discovered that all the currently existing maps in Carla are custom maps that have no correlation to real locations, and as such, it would be nonviable to geo spoof location data to the android head unit as the Carla locations are not real. Source: https://carla.readthedocs.io/en/latest/core_map/#generating-a-map-navigation

  As such, it would be best to reorder the sequence of tasks to be completed and instead of tackling the android head unit first, it would be better to begin developing custom maps that are based on real life locations as mentioned in prior meetings. As for the work to be done today, I will (for the meantime) look into connecting the shifter to the current system and integrating it with the other peripherals in the Carla bench.
- Suppressed preexisting code left by Carla developers that only allow for one joystick to be connected (The wheel + pedals)
- Initialized an additional joystick in 'testbed.py' using:

  ```
  self._joystick1 = pygame.joystick.Joystick(1)
  self._joystick1.init()
  ```
- Copied over previous capstone team's shifter functionality from 'client_MODIFIED.py', where shifting into second gear acts as drive, reverse acts as reverse, and any other gear acts as neutral. This is because the vehicle's in the Carla simulator are all automatic.
- Following manual testing, it was concluded that the shifter was integrated successfully and exhibits the same behavior as it did in the windows version of the system.

- Did some research and found that Carla has options to use a manual gearbox. It would make sense to integrate that functionality into the shifter controls since as an automatic, the shifter is essentially redundant.

- Tested the preexisting keyboard controls for using the manual gearbox in the Carla simulator, which can be used by pressing the '**M**' key. To shift up and down, use the '**.**' and '**,**' keys respectfully.

- Next step is to map the entire gearbox to different gears in the shifter. In order to do this, I first need to see what button index each gear in the shifter is mapped to through manual testing.

- **The following was printed to the terminal window after shifting through the gears in sequential order:**

```
event.button = 0
event.joy = 1

event.button = 1
event.joy = 1

event.button = 2
event.joy = 1

event.button = 3
event.joy = 1

event.button = 4
event.joy = 1

event.button = 5
event.joy = 1

event.button = 6
event.joy = 1

event.button = 7
event.joy = 1
```

The 'event.joy' line tells us that the input is from the joystick at index 1 (the shifter) and the 'event.button' line tells us what button index each gear in the shifter is mapped to. As shown, every gear 'n' is mapped to the button 'n-1', except for the reverse gear, which is mapped to button index 7. The next step is to program a mechanism which maps the shifter gears to the in-software gears as currently, I only have the ability to shift up and down opposed to shifting to individual gears.

- After looking through and reviewing existing code, the lines used to shift up and down were found to be '**self._control.gear = self._control.gear + 1**' and '**self._control.gear = max(-1, self._control.gear - 1)**' respectively. Ctrl + f: K_COMMA to find corresponding lines.

- Implemented a flag named '**globalManualFlag**' which is used to check whether the transmission is in manual mode in the global scope of the program. Next steps involve building the manual shifting mechanism.

- The manual shifting functionality was added into the '**parse_events**' function and is now working as expected. The manual shifting functions will undergo manual testing to see if any notable issues arise.

- It was found that the issues where the speed and rpm values switch on the Arduino side can occur even when a new program is not uploaded. As such, code was added to the Arduino program to detect whether this error had occurred and swap the values.

- Currently, it is difficult to test the full range of speed and rpm values on the current map. To resolve this, I will look into how to load different existing Carla maps into the current system as the program currently loads a default map from the server side.

- To change the map being loaded, the statement '**sim_world = client.get_world()**' in the **game loop** section of 'testbed.py' was changed to '**sim_world = client.load_world('Town06')**'. The map **Town06** was chosen as it consists of predominantly straight roads, allowing me to test the speedometer and tachometer more thoroughly as I shift through the gears in the manual mode.

- It was discovered that after approximately 150 kph, the speedometer became less accurate and overestimated the speed of the actual vehicle by upwards of 20kph. In order to remedy this issue, code will be added to the Arduino program to check if the speed is above 150, and if so, adjust the scaling factor to a new one. The new scaling factor will be obtained through trial and error.

- The scaling factor that produced the most accurate results on the speedometer after 150 kph is: '**mappedSpeed = map(incomingSpeedValue,0,120,0,155);**'. This scaling factor is accurate all the way from 150 kph to 240 kph, which is the maximum speed on the speedometer.

- Through manual testing, it was also discovered that the speedometer acts frantically at lower speeds, particularly at speeds under 25 kph. This was discovered now since we now have the ability to manually shift into a high gear, allowing the vehicle to creep forward very slowly, whereas before the automatic transmission would shift on its own, causing the vehicle to speed up quickly. In order to remedy this issue, the same tactic will be employed as before, by adding code to the Arduino program to detect our speed and apply a different scaling factor.

- The best scaling factor for speeds under 25 kph is '**mappedSpeed = map(incomingSpeedValue,0,3200,0,155);**'
- Upon further manual testing, it was found that speeds over 220 kph aren't represented as accurately as they could be on the speedometer. To improve the accuracy of the gauge cluster at speeds past 220 kph, an additional scaling factor was added to the Arduino program, that scaling factor being '**mappedSpeed = map(incomingSpeedValue,0,123,0,155);**'.
- After adding this system of variable speed scaling in the Arduino program, the speedometer's behavior is much more accurate to the in-simulator car's speed when shifting through the gears using the manual transmission mode..