RIDWAN HOSSAIN
100747897

2022/05/26

- Looking into how to send multiple values at once to the Arduino board without experiencing the current issues present

- Was able to send the speed and engine rpms values to the arduino board successfully, using the modifications shown below:

  Arduino side:
  String str1 = Serial.readStringUntil('\n');
  String str2 = Serial.readStringUntil('\n');

  Python side:
  ser2.write(bytes(f"{str(int(speed))}\n", encoding='utf-8'))
  ser2.write(bytes(f"{str(int(engine_rpm))}\n", encoding='utf-8'))

  Essentially, the method was to send the integer values as strings, but wrap them as bytecode using utf-8 encoding, as that is the encoding format that the arduino serial port uses. This allowed the data to be properly read as strings on the Arduino end of the system and the results provided upon adopting this method are as they should be.

  Sources used for above method:
  1. https://stackoverflow.com/questions/48373982/pyserial-sending-and-receiving-multiple-data
  2. https://stackoverflow.com/questions/35642855/python3-pyserial-typeerror-unicode-strings-are-not-supported-please-encode-to
  3. https://stackoverflow.com/questions/19511440/add-b-prefix-to-python-variable

- Next step is to convert the strings into integer values on the Arduino side and then map those values to analog signals as I have done before.

- **Note:** The Arduino board uses a Little Endian structure, so the order of values sent to the board will be received in the reverse order on the Arduino side.

- Was able to convert the string values to integers and send them to the gauges as analog signals successfully. Also increased the baud rate on all ends to **2000000** to improve gauge responsiveness. Next step is to acquire a correct scaling factor for the rpms to analog signals through trial and error.

- **Note:** Discovered that after each upload of a new program to the Arduino board may result in the board switching the order in which it receives the speed and rpm values. Keep note of this. An easy way to fix it is unplug and plug in the Arduino board.

- I'm finding that the formula currently used to retrieve the engine's rpm values are not that accurate. The rpm values do not change often and are very rough. I'm looking into other methods to retrieve the vehicle's engine rpm values in Carla, which may be difficult as there is no accurate way to do it without going into the Unreal Engine level.

- **Note:** In the case I want to look into how to extract the engine's current rpm values from Unreal Engine 4, I may use the following source as an example: https://github.com/carla-simulator/carla/discussions/4197
  However, this is unlikely as I would have to then learn unreal engine, too far off task.

- Looking into how to obtain wheel rpm from speed, then engine rpm from wheel rpm. There is some useful information in this resource: https://www.omnicalculator.com/everyday-life/rpm

- **RPM Calculation Notes:**
  - To convert vehicle speed to a wheel's RPM, use the following formula:
    - $$Wheels\ RPM = \frac{Vehicle\ Speed\ in\ mph}{tire\ diameter * \pi * 60/63360}$$
  - Engine RPM = Wheels RPM × Transmission ratio
  - The Carla python interface allows the user to obtain a vehicle's transmission ratio based on the current gear, thus, this method of obtaining is plausible. The Carla interface however does not allow the user to access a vehicle's wheel diameter, so as a result, I will be using a single value and apply them to all vehicles as it will at least allow me to obtain a rough value of the RPMs. The tire diameter I will use is 64cm and was obtained from this source: https://github.com/carla-simulator/carla/issues/135

- Implemented the new method of RPM calculations onto the python side and printed the values to the terminal for inspection. The values look substantially more accurate than the previous method and also change based on the shifting of the gears as well as the transmission ratio and speed. Next step is to send these values to the Arduino side and map them to analog signals.

- The RPM values are being received on the Arduino side and a scaling factor must be acquired to obtain a correct scaling from the software numbers to the analog signals. This scaling factor, like before, will be obtained through trial and error.

- Through trial and error, it was found that the best scaling factor for the RPMs are: **'int mappedRPM = map(incomingRPMValue,0,3400,0,155);**'. It must also be stated (for the record) that the best scaling factor for the speed was found to be '**int mappedSpeed = map(incomingSpeedValue,0,115,0,155);**'.

- This concludes the work that needs to be done on the gauge cluster. The next step is to begin working on the android infotainment system.