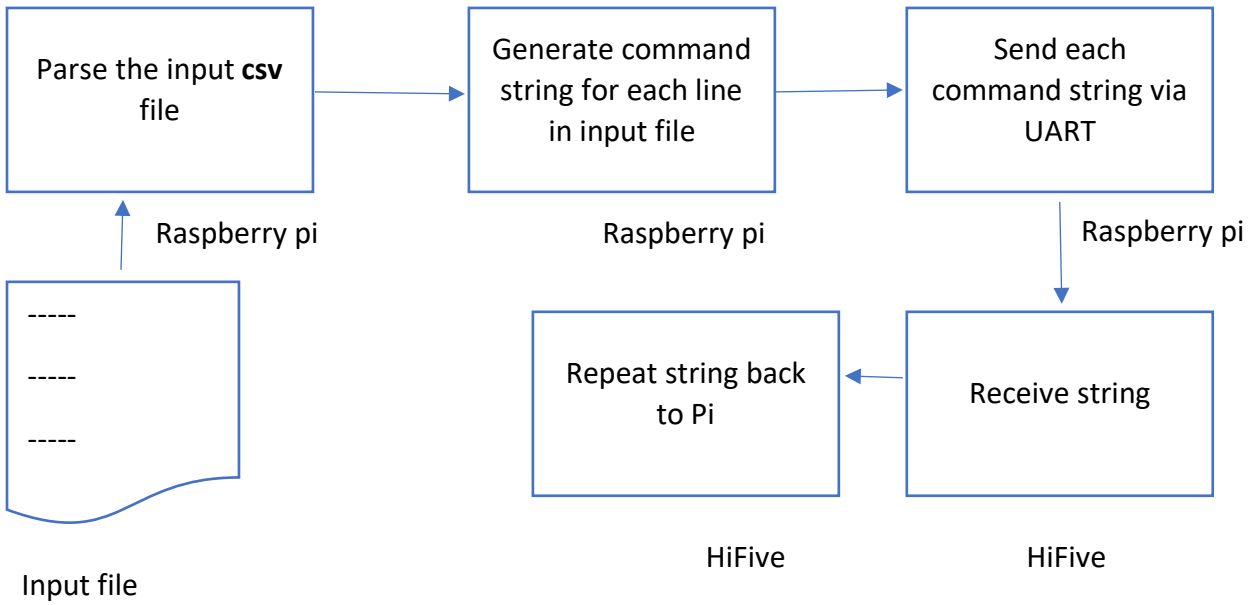
Milestone-2

Instead of using terminals, you will now create a python program on the Pi to communicate with the HiFive. Your task is to send formatted command strings to the HiFive that can control the steering angle, speed and duration of a command. These commands will be supplied from a **csv** file on the Pi. In this milestone, you will implement the necessary code for parsing that input file and sending the correct values to the HiFive from the Pi.

The following flowchart describes this milestone:



A "command" consists of a speed value and steering angle value that the car should be set to, and a duration value for how long to hold that configuration. The commands in the input csv file will have the following format:

<angle value>, <speed value>, <duration>

- The angle value will be an integer between –45 and 45.
- The speed value will be 1, 2 or 3 for driving forward (at different speeds), -1, -2 or –3 for backward and 0 for stop.
- The duration will be the time, in *seconds,* for which that command should run.

For example:

30, 2, 3

The command above means the car should steer to 30 degrees, move forward with speed flag 2 and hold this configuration for 3 seconds. To read in the input file you can use python's **csv** library. Here is a tutorial on how to use the csv library, but there are many more out there, so feel free to explore!

After reading these values in from the input file on the Pi, you will generate your command strings containing the information you collected to be sent to the HiFive. The actual format of the command strings is up to you; you can send them as is (i.e., as a string containing three values separated by commas), add additional text to indicate each parameter or you can use the sample format you will find below.

After generating a command string, you will need to send it via the serial connection (UART) to the HiFive from the Pi. At the HiFive end, you will need to read in these command strings being sent from the Pi and repeat them back to the Pi using a different UART connection (similar to what you did in lab 9).

The following *pseudo-code* provides a general idea of the code you will need to write (*it is NOT actual valid python code*):

```
open a serial connection to /dev/ttyAMA1
open csv input file
for each line in input file:
      read angle value, speed value and duration value -> angle, speed, duration
      command_str = "angle: " + angle + "speed: " + speed + "duration: " + duration
      Write command to /dev/ttyAMA1
close serial connection
```

To achieve the functionality from above, you need to use Python's **pySerial** API which can be used by importing the **serial** package. You may use this link to a simple tutorial of the pySerial:

```
import serial
```

With pySerial, you should create a serial channel for writing to the HiFive over **/dev/ttyAMA1**. Note that the channel should be opened with the baudrate **115200** bps.

```
ser1 = serial.Serial(...)
```

The command strings you generate can then be sent to the HiFive by using the serial write function:

```
ser1.write(command_str)
```

However, the **write** function requires a byte value, which must be formed from whatever type your command variable is in order to send it to the HiFive.

```
ser1.write(bytes(command_str))
```

Finally, after all of commands are sent, the serial connections can be closed by invoking the serial close function:

```
ser1.close()
```

Tasks for Milestone 2

1. Wire up the HiFive and the Pi the way you did for lab 9.
2. On the HiFive, use the same program you wrote for lab 9 (for communicating between the HiFive and the Raspeberry Pi). Build and upload it to the HiFive to get it ready to receive the command strings from the Pi.
3. On the Pi, create a python source file called **`prog.py`** and write your python code inside this file.
4. Download the input csv file from Canvas. It is inside the file name **milestone2.tar.gz.** You will need to read this file in your python code to get the commands.
5. Open a new terminal window and run the **screen** command which receives characters (i.e., run screen with /dev/ttyAMA2), run your python program and demonstrate the transferring of information. Python programs can be executed on the terminal as follows (assumes program is named *prog.py* and that you are in the same directory as the .py file):

   ```
   python prog.py
   ```

Submission:

You should submit the written python code on Canvas as submission for Milestone-2.