EECS 388: Embedded Systems Lab

Final Project

Project Description Manual

Objective:

In this project, you will have the opportunity to apply the knowledge and skills you have acquired in the previous labs. The project centers around designing a simple embedded system that encompasses automatic braking and automatic steering functionality for a self-driving car. You will utilize the embedded components (serial communication, sensors and actuators) that you have encountered so far to create a prototype that is capable of distinguishing between different braking scenarios (obstacles which are far away, near or very near) and is also able to simulate automatic steering based on a video input using a pre-trained DNN (Deep Neural Network) model.

Project Summary:

1. **Team Composition:** This final project is designed for groups of **2 students**.
2. **Total Marks:** This project will be evaluated out of **100 marks**.
3. **Project evaluation:** Your project will be evaluated based on the following parts:
    a. **Part 1 (75 marks):** Part 1 comprises of three tasks. Each task carries 25 marks.
        i. **Task 1:** This task involves using the TFmini Lidar sensor to collect distance measurements for objects within your surroundings. Drawing from your knowledge gained in Labs 02 to 04, you will modify the provided **comm.c** to address distinct braking scenarios.
        ii. **Task 2:** For this task, you will execute the **dnn.py** script from Lab 08 to utilize a pre-trained DNN model. This model uses video frames (available in the provided directory) to infer the steering angle. Your challenge is to adapt the **dnn.py** script to transmit this steering angle to the HiFive board using the board-to-board UART connection knowledge you have acquired in Lab 09.
        iii. **Task 3:** In this task, you must use the received angle values to drive a servo motor. Upgrade your **comm.c** file from Task 1 and write the code for driving the servo using the angle values received from Raspberry Pi. Use your understanding of Lab 05 to implement this part.

    b. **Part 2 (25 marks):** This part requires you to prepare a short presentation, up to 10 minutes maximum, where you talk about the following aspects:
        i. Your understanding of the tasks
        ii. An explanation of how your code works
        iii. Any issues encountered while working on the project and how you overcame them

Project Deliverables:

1. **Code:** all members of the team should submit the same **comm.c** file and **dnn.py** file to Canvas. Without the code submission, we will be unable to grade you. Submit the files directly, do not zip them.
2. **Demo:** A demonstration of all the components working together and producing the expected output
3. **Presentation:** A short presentation, up to 10 minutes maximum.

Deadlines:

The deadlines for the two parts are given below:

- **Part 1:** The demo for Part 1 is due within 3 weeks from the start of final project (the start for each team is their respective lab sessions). The demo can be given any time within this period if the team has completed their project.
- **Part 2:** The presentation for Part 2 is due in the 3rd week from the start of the final project. The presentation cannot be given earlier than this. The day of the presentation is also the last day for the demo.

We recommend that teams get their code demos done ahead of time so that there is no rush on presentation day. There will be no extra time given to setup and demo during the presentation.

Task Details:

There are two compressed source files for the final project. The **HiFive** folder is the source code meant to run on the HiFive and the **DNN** folder has the source code which needs to run on the Raspberry Pi.

Task 1:

Task 1 asks you to implement a prototype automatic braking system using the TFmini lidar sensor from Lab 04. The idea behind this task is to measure the distance between our sensor (and therefore our hypothetical car) and any obstacles ahead in the direction of the sensor and decide if brakes need to be applied and how hard.

For the purposes of this project, we define any object that is further than 200 cm as a safe distance away and no braking is needed. If any object is less than 200 cm but more than a 100 cm away, we assume that it is getting close enough to warrant braking, but only **lightly**. For anything less than 100 cm away, we assume that the object is getting too close to the vehicle, and the brakes need to be applied in full force. Finally, for something that is less than 60 cm away, we assume it is too close to us to keep the vehicle in motion and so it needs to stop and be stationary. To simulate braking, you will change the color of the LED to indicate the level of braking. The levels of braking, and their representation using LED colors, are detailed below:

Distance	Action	LED Color
> 200 cm	Safe distance, no braking	Turn ON Green LED

100 cm < d ≤ 200 cm	Close, brake lightly	Turn ON Yellow LED (Red+Green)
60 < d ≤ 100 cm	Very close, break hard	Turn ON Red LED
d ≤ 60	Too close, must stop	Turn ON *flashing* Red LED (Flashing period = 100ms)

To implement this, you will need to modify the **auto_break()** function in the source code. In this function, you should get the distance readings from the TFmini, parse it, and, using the distance value, display the correct color on the LED. The connection for the lidar will be similar to lab 04. Keep in mind though, the lidar should be using **UART 0**.

Task 2:

To simulate our car's automatic steering, we will use the DNN model from Lab 09. If you recall, the DNN model predicted the turning angles in degrees based on the input video footage. The script **dnny.py** runs the inferencing using the pretrained model with the video frames as input and produces a degree value as prediction and stores this in a variable called **deg** in the python file.

It will be your task to modify **dnn.py** so that the python script sends this predicted angle value to the HiFive from the Pi using the UART connection, instead of simply printing it to the terminal.

To initiate a serial (i.e., UART) connection from a python script, you will need the **serial** library. That has already been added to the **dnn.py** file. You would first need to instantiate a new serial connection and store it in a variable. A new connection can be created using the following command, and stored in a variable called **ser**, like so:

```
ser = serial.Serial("<path to connection file>", baudrate)
```

The path to the connection file is the ttyAMA*x* (where *x* is 1 or 2) file that you used in Lab 09. We recommend you use ttyAMA1, although the project can be accomplished using ttyAMA2 as well. The *baudrate* is the same as what you had to use in the Lab 09 **screen** command.

Once established, you need to actually write to that connection, so that it can be sent by the Pi and received by the HiFive. To do that, you can use the **write()** method of a serial object. For example:

```
ser.write()
```

Keep in mind, however, that passing any python objects directly might not give you the correct output on the other end. It is a good idea to first convert your input to **write()** into a byte format. You can do that using the **bytes()** function in Python. You can run the **dnn.py** file the same way you did in Lab 08.

Task 3:

For task 3 you will need to connect the HiFive with the Pi via UART so that they can talk to each other. Use the information given in Lab 09 to wire up the HiFive and Pi; remember you only need the HiFives UART1 to connect with the Pi. UART0 will be used for the lidar.

Once connected, you need to modify your **comm.c** file to receive any incoming bytes from the Pi and extract the angle value. Once again, your Lab 09 code here will help, but you may need to do some extra steps. You can, for e.g., use the **sscanf()** function to extract the data, however this is only a suggestion and there are many other ways to go about it. Here is more info on **sscanf**:

- https://en.cppreference.com/w/c/io/fscanf
- https://www.tutorialspoint.com/c_standard_library/c_function_sscanf.htm

After successfully extracting the angle value, you must then pass it to the **steering()** function, which will turn the servo motor. To implement the steering function, you may look at your Lab 05 code.

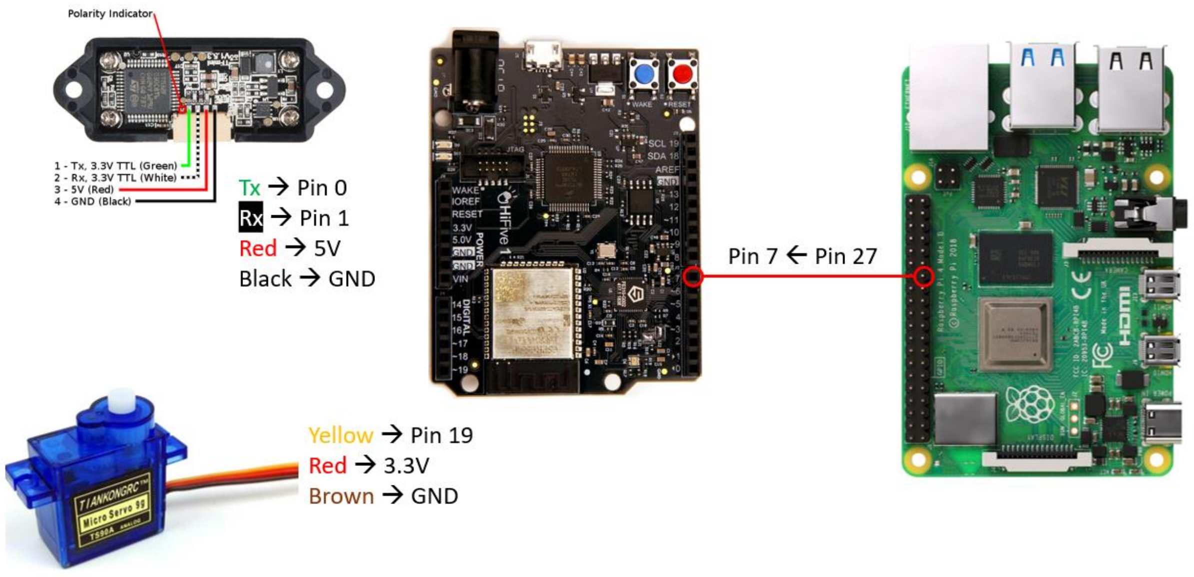
With all this done, you should have a simple prototype for automatic braking and steering!

For the demo, you will be asked to show that your HiFive LEDs show the right color based on distance from some object, that your **dnn.py** script is able to send the angle predictions to the HiFive over UART and that your servo motor responds accordingly.

<u>Necessary Equipment:</u>

1. HiFive Board
2. TFmini Lidar Sensor
3. Servo Motor
4. Raspberry Pi

<u>Connection Diagram:</u>


Polarity Indicator
1 - Tx, 3.3V TTL (Green)
2 - Rx, 3.3V TTL (White)
3 - 5V (Red)
4 - GND (Black)
Tx → Pin 0
Rx → Pin 1
Red → 5V
Black → GND
Pin 7 ← Pin 27
Yellow → Pin 19
Red → 3.3V
Brown → GND