EECS388 Final Project – Spring 2021

Self-Driving Car

In this project, you are going to implement a self-driving car by incorporating elements from the previous labs. You are also going to use a new board (PCA9695) for driving servo and DC motors. PCA9695 uses I2C to receive commands from HiFive board to generate PWM signals for the servo and DC motors.

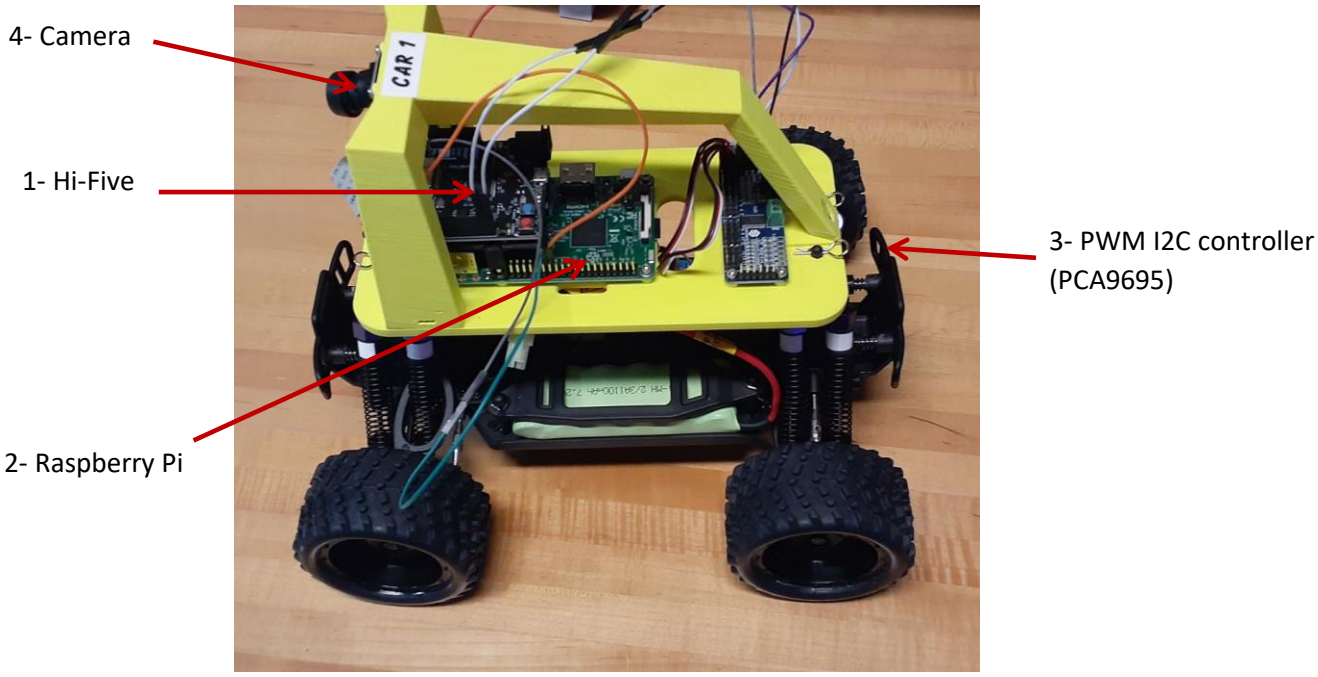


Figure 1: Self-driving car prototype

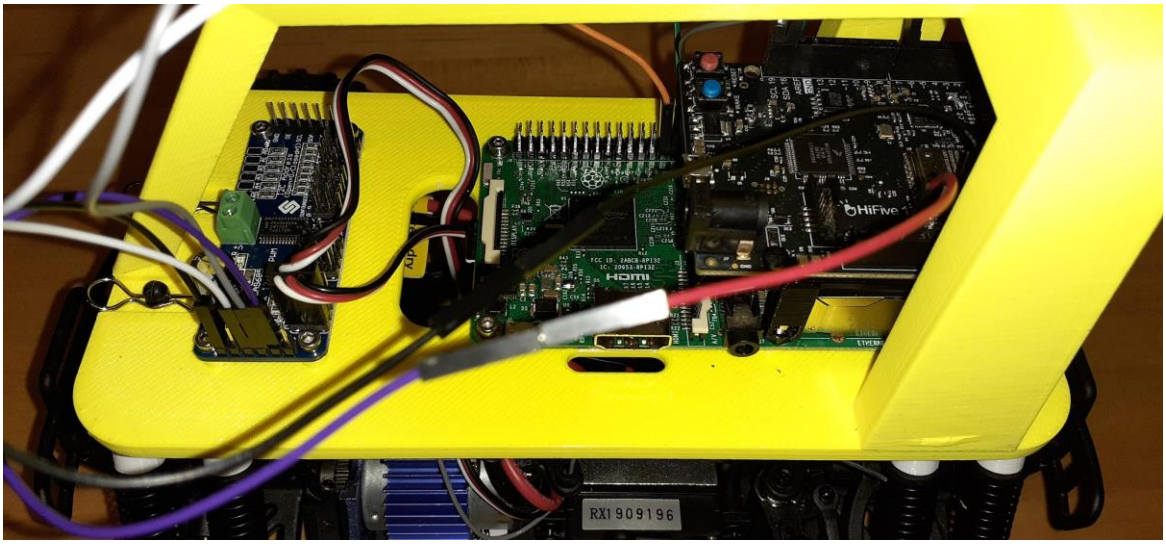


Figure 2: HiFive, Pi, and PWM I2C controller boards

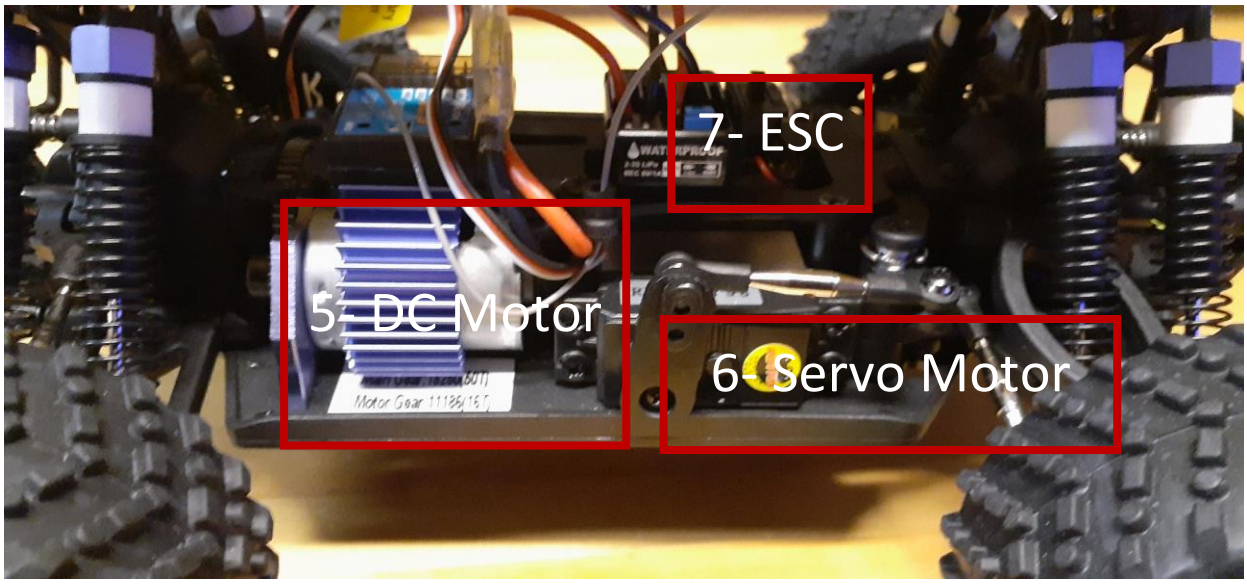


Figure 3: DC motor, Servo motor, and ESC on the car

Figure 1, Figure 2, and 3 show the car prototype that you are going to use. It has seven main components:

1. HiFive
2. Raspberry Pi 4
3. Motor driver (PCA9695)
4. Camera
5. DC motor
6. Servo motor
7. Electronic Speed Controller (ESC)

This project is split into four milestones:

For **Milestone 1**, your goal in this project is to first use the HiFive board to send I2C commands to PCA9695 to drive the servo motor (for steering).

For **Milestone 2**, finish implementing HiFive for the DC motor control (moving forward). Then connect the Pi to the HiFive board using UART like lab 7. This is to set up a connection between the two boards for sending steering commands from the Pi to the HiFive board.

For **Milestone 3**, you will modify the python code to use the camera (instead of processing video frames), run the DNN inference engine on captured images, and then send the steering commands from the Pi to the HiFive board (using UART) which will control the motors (using PWM I2C controller).

Optionally, you can develop this project further for extra credit. You can find more information about the extra credit when the **Milestone 4** section is released. The maximum extra points that you can get is 20% of the total final project grade. Before you start working on any extra credit project, you should discuss it first with your GTA to evaluate its feasibility and the number of extra points that you can get.

You can check the deadline for each milestone in Table 4.

Milestone	Description	Deadline
1	Drive motors using PWM	Apr 23rd
2	Connect HiFive and Pi	Apr 30th
3	Self-driving car	May 7th
4	Extra credit (open ended)	May 7th

Table 4: Deadline for each milestone

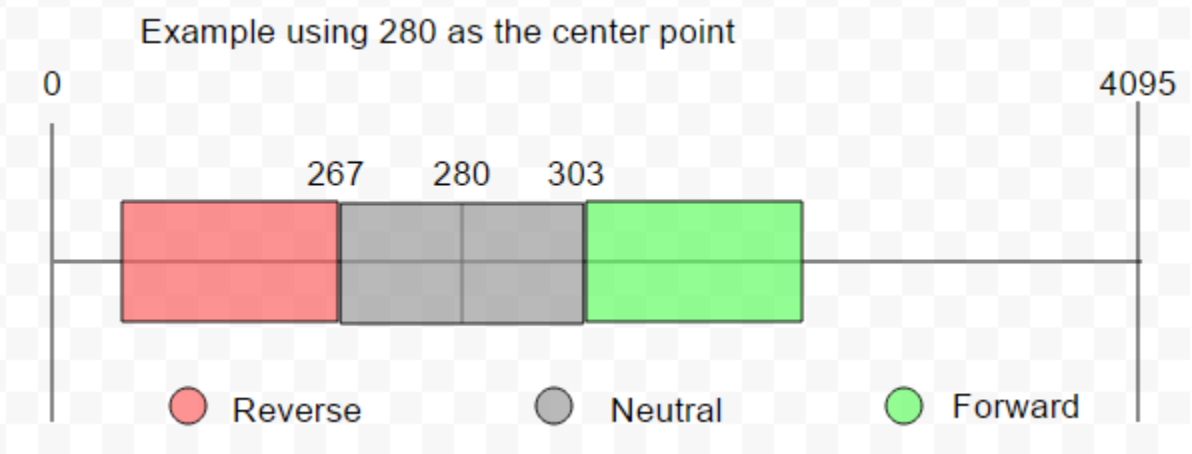
Milestone 2

There are two tasks to be completed for this second milestone: (1) implement the function to drive the RC car forward, (2) implement UART communication between the Raspberry Pi and the HiFive board to allow the inferenced angles outputted from the DNN to be inputted into the steering function.

Part 1: Drive Forward

Controlling the motor is the exact same as controlling the steering servo with one key exception; the motor is connected to an Electronic Speed Controller (ESC), which has a few quirks. For starters, the ESC must be configured before it will move - this process is relatively simple - you must write an arbitrary PWM signal in cycles for the ESC to calibrate. On this prototype, the calibration is accompanied by a beep (technically the calibration beep is the second beep after turning on the power switch) signaling calibration being completed. From this arbitrary point, there is a dead band of plus or minus 23 cycle lengths, sending a PWM outside this dead band will make the motor drive forward or backwards respectively. This relationship is illustrated below:

Be Advised: these RC cars are designed to drive at very high speeds. Keep the values between 265-305 and be sure the robot is propped up so it can't suddenly drive away!



Task 1: Calibrating and defining the stop function

First, you should add in your implementations for the breakup, write, and steering functions from milestone1 into the new version of eecs388_i2c.c for milestone2.

You will implement the following function in order to stop the wheels from moving. This will be accomplished by setting the LED0's duty cycle to 280. Also, this will cause the ESC to calibrate itself when first called and with a two second delay.

```
void stopMotor(){
	/*
		Write Task 1 code here
	*/
}
Example Use: stopMotor(); -> sets LED0_Off to 280
```

Task 2: Drive Forward function

You will implement the following function in order to make the wheels drive forward. Further details are provided below as well in the comments in the c file.

```
void driveForward(uint8_t speedFlag){
	/*
		Write Task 2 code here
	*/
}
The given speedFlag will alter the motor speed as follows:
speedFlag = 1 -> value to breakup = 303
speedFlag = 2 -> value to breakup = 305
speedFlag = 3 -> value to breakup = 307
Example Use: driveForward(3); -> sets LED0_Off to 307
```

Part 2: Steering RC from DNN

You will now combine work done in the previous milestone with Lab 7. After completing this milestone, the DNN running on the Raspberry Pi should process the video file "epoch-1.avi" frame-by-frame, the predicted angle for each frame should be sent to the HiFive board using the UART serial communication setup in Lab 7, and finally that angle should be applied to the steering function implemented in Milestone1.

You should first add in the serial code you wrote for Lab 7 to the eecs388_i2c.c file (for the function `raspberrypi_int_handler`) as well as add the modifications to the python file dnn.py.

To use the DNN for steering, you need to use create a “drive loop” in the main function (in eecs388_i2c.c) such that the prediction angle for each video frame is used to control the car’s steering (that is, the HiFive should loop between receiving an inferred angle from the Pi and calling the steering function with the angle). In order to do this, you will either need to convert the received value from the Pi via UART into a useable integer value or pre-modify the value in the Python program.

If you choose the former approach, here are helpful functions in the c standard library to achieve the above task. These are `strncmp` and `sscanf`.

```
char * lhs = "Hello World!";
char * rhs = "Hello";
int count = 5;
// strncmp returns zero if lhs == rhs for the first count amount of chars
// This would return 0 since the first five characters are the same
int ret = strncmp(lhs, rhs, count);

// sscanf can be used to take numbers out of strings
// for example, we can get the int value 388 out of the following string
char * str = "Number:388";
int val;
// This assigns the int 388 to val
// The +7 indicates that the int starts at the 7th character in str
sscanf(str+7, "%d", &val);
```

To **demo** your code: You should have the HiFive first calibrate the motor, drive forward for 2 seconds, stop the motors, and then enter the loop for steering demonstration (at this point you can start the python script for DNN inferencing).

For **submission**, submit a tarball (only one per group) of the HiFive directory containing the modifications made. You **do not** need to submit the Raspberry Pi directory as the code is from a previous lab.