Memo

Team: Team 09
Date: 4/20/23
Subject: Final Project Testing Plan

Required Materials

Hardware:

- Breadboards for controllers, steering, and misc, and their respective connections
- PID controller with backup “watchdog” controller
- Relay boards
- DAC boards
- DC-DC converter
- Polycase enclosure
- 3D printed fittings for the enclosure
- Screws to attach backup parts to the printed fittings
- Microcontrollers (one Arduino Nano and two ESP32-S3s)

Software

- Simulink Scripts
  - PID implementation on controller
- C code scripts for main and watchdog controllers
  - ESP-IDF libraries
  - Configurations for UART, I2C, and other serial communication
- Arduino code scripts for auxiliary controller
  - Adafruit ISM330DHCx Arduino library
  - Arduino IDE

Other

- Demonstration vehicle with four motor-controlled wheels
- User-interactive input box: steering wheel and throttle pedal

Setup

Enclosure 1 (auxiliary controller section):

The auxiliary controller (Arduino Nano) and sensors (accelerometers and potentiometers) are sitting on top of each other via solderable breadboards, wired to their respective relay boards. All the hardware parts are mounted together and organized to fit within our 8in x 8in x 8in Polycase packaging.

Enclosure 2 (main controller section):

The main and watchdog controllers (two ESP32-S3s) are sitting on top of each other via solderable breadboards, wired to their respective relay boards. All the hardware parts are mounted together and organized to fit within our 8in x 8in x 8in Polycase packaging.

Demonstration vehicle:

The physical car prototype with its wheels and motors are sitting atop an elevated surface where the wheels are suspended in the air. The car is wired to the output of the main controller enclosure. The input to the auxiliary controller is connected to the user-interactive steering and throttle input demonstration parts, which incorporates a steering wheel and a throttle pedal.

Pre-testing setup procedure

Enclosure 1 (auxiliary controller section):

Ensure that the software code for the auxiliary controller is uploaded onto the Arduino Nano controller. The code files can be found in our Github repository, under the *SrDes-arduino_aux_controller* folder. Since this is an Arduino-based program, make sure to have the latest version of the Arduino IDE installed, and then add the *Adafruit ISM330DHCX* library into the Arduino IDE. Once finished, compile and upload the Arduino code to the controller. To ensure no errors, the serial monitor can be opened up, and the raw data signals from the sensors should be polling in real-time.

Enclosure 2 (main controller section):

Ensure that the software code for the main and watchdog controllers are uploaded onto the respective ESP32-S3s. The code files can be found in our Github repository, under the *SrDes-main_controller* and *SrDes-watchdog* folders. Since this is an ESP32-based program, make sure to have the latest version of the ESP-IDF libraries installed, and then run the following commands in the terminal for each directory:

- Idf.py set-target esp32s3
- Idf.py build
- Idf.py -p [PORT] flash monitor (replace PORT with user's port number connected to their ESP32-S3 chip)

If there are no compilation errors detected within those steps, the software has been successfully uploaded to the ESP32-S3s, and any print statements for the data values/calculations should appear on the terminal window.

Demonstration vehicle:

No code is needed to be pushed to these components. Ensure that the demonstration vehicle is properly sitting on top a flat surface and wired to the main controller output channels, and that the steering and throttle input mock-up is also sitting on the same surface and connected properly to the auxiliary controller input channels.

Testing Procedure

Enclosure 1 (auxiliary controller section):

We first walk through the composition of all the components within the packaging box and the organization of them together with the wiring. Next we show that the values coming from the Adafruit accelerometer are realistic. By moving the entire packaging around in different directions (up and down, left and right, side to side, etc), we can see the raw values of the data being collected in real-time on the Arduino serial monitor. The order of the data values should be the acceleration values in the X, Y, and Z directions, the steering input, and the throttle input (the latter of two which can be tested by turning the knob to vary the signal magnitude).

Enclosure 2 (main controller section):

We also start with a tutorial explaining all the parts and connections inside the packaging. Through the terminal output within the main controller, we want to verify that the sensor readings from the auxiliary controller are transmitted to the main controller simultaneously over UART with the same order as mentioned above. We can also demonstrate the watchdog controller being triggered by manipulating the input signal so that the main controllers begin to misbehave and product results inconsistent with our design. Upon reaching that point, the relay switches attached to each microcontroller will make a “click” sound, followed by their on-board LED light getting turned off. We will also show results from the Simulink model displaying how much power would ideally be assigned to the left and right motor controllers as a mode of validation.

Demonstration vehicle:

We explain the parts that make up the demonstration vehicle, including its four motor-based wheels. We also showcase our new user-interactive feature to play around with steering and throttle input.

The user can manipulate the steering input by turning the wheel to the left or to the right, and throttle input can be achieved by pushing down on the throttle pedal. Based on varying the inputs to the steering and throttle parts, the demonstration vehicle should respond to the user inputs accordingly by adjusting the motor power distributed to each of the four wheels in an appropriate and expected manner.

Measurable Criteria

Enclosure 1 (auxiliary controller section):

For our Arduino Nano auxiliary controller, we can show that the inputs which are being sent to the microcontroller are properly being measured, and can be sent to the main controller esp32s3 via UART. Additionally, we will measure the response time between the arduino

Enclosure 2 (main controller section):

For our main controller, we can show that the controller takes in the inputs to the system (steering angle, throttle, and x-y-z acceleration values). The main controller will then use these values to calculate the linear speed of the car. The steering angle, throttle, and linear speed of the car are integrated with the Linear Model (which we developed using Simulink) to calculate how much power to give each of the two motors (left and right). We'll demonstrate two different versions of the code, one with the full PI model and one with a simplified version of the code to simply demonstrate what it would look like on the car.

Demonstration vehicle:

We want to illustrate the accuracy of how the user's inputs to the steering wheel and throttle pedal can be modeled on the physical 3D vehicle. The user should be able to successfully turn the steering wheel one way or the other and add throttle input without any issues. In turn, the demonstration vehicle should respond to the user's inputs almost immediately where we can justify the car's motor output and overall resulting behavior.