University of Pittsburgh
Department of Electrical and Computer Engineering
ECE 1895: Junior Design Fundamentals
Dr. Samuel Dickerson

Design Project #1: Line Following Race Competition

Introduction

For this project, you will design a robot with the ability to follow a line and compete against your classmates in a race. Although it may sound simple, having a robot race around a line is very challenging and requires that you utilize everything that we have learned thus far. In modifying your robot design, you will add several new features to your robot , including

- Edge-triggered interrupts to detect collisions
- Hardware timer modules to drive motors
- A more robust FSM controller that can overcome obstacles

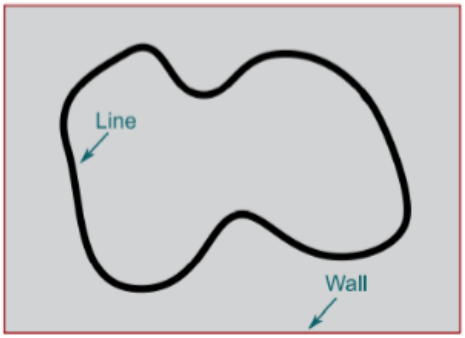

Figure 2. Create a robot explorer that follows a line.

Your design will be evaulated across several metrics, including

- **Speed** – The amount of time it takes for your robot to complete the track
- **Computational Efficiency** - The amount of memory your design requires
- **Power Efficiency** – The amount of energy required for your car to operate

Preparation

- For this assignment, you will need
    - A fully assembled **TI-RSLK Max Kit** 
    - A race track
    - A team of 3 engineers

Assignment

Completing this project will require the completion and integration of several new software components. You will work in a team of 3 engineers, and the team can decide which member is responsible for which component. However, each team member **must assume sole responsibility for at least one part of the final, completed design**.

Motor Drivers

For your first design, you use SysTick timers to control the motor drivers. However, this is inefficient and may also become problematic if too many resources are trying to use SysTick. You must modify your design such that the motors are controlled by the Timer A hardware module being used to generate PWM signals. Complete lab module 13 to learn now to do this.

Collision Detection

You will improve your robot's ability to detect collisions by making use of edge-triggered interrupts on the bump sensors. Complete lab module 14 to learn how to do this.

Finite State Machine

When we first introduced finite state machine control, the requirements were simple and the speed was not considered variable. Recreate a new FSM controller that leverages your modifications and is robust to conditions and scenarios your robot will encounter. Refer to TI-RSLK Module 7 - Finite State Machines for additional help.

If you have a team comprised of 4 people, then you must add one additional feature from among the following

Graphical Display

Your robot does not have the ability to convey information outside of LEDs, which makes debugging in real time difficult to do. One solution is to add a graphical display. You must interface to an LCD screen that provides real-time updates of the state of the robot. Interfacing to an LCD is explained in TI-RSLK Module 11 - Liquid Crystal Display (LCD)

Data Logger

Another useful debugging technique is data logging. The difficulty with implementing data logging is the availability of memory, which can run out quickly. To solve this problem, one can utilize the expanded storage flash memory provides. You must interface to the on-board flash storage and implement a data logging protocol. Reading and writing to flash memory is explained in TI-RSLK Module 10 - Debugging Real-time Systems

*You may use the project **JACKIFSM** as the starting point for this project.*

Competition

For the competition, you will be provided a race track, you're your submission will be evaluated across three metrics: Speed, Memory and Power. All points listed below are for ranking purposes, not grading. To be eligible to win, the code executing on your robot when it races must include completed versions of all the previously described modifications to the robot.

Speed Trial

The first, and most important objective, is to get your robot to go around the track as fast as possible. Your robot must be able to make at least one complete lap, without intervention, to have the time recorded. You may have 3 attempts around the track and your best time will be recorded.

- 1st Place – 9 points (fastest time)
- 2st Place – 6 points
- 3st Place – 3 points

Code Efficiency

The next metric of interest evaluates the efficiency of your code. We can measure the efficiency of code by the amount of memory it consumes. This can be measured easily in Code Composer Studio using the Memory Allocation viewer tool.

Memory Allocation View

How much memory am I using?

The tool shows the percentage of memory used by each memory unit and it also indicates the number of bytes allocated. We will use the sum total of the # of bytes allocated across all memory modules as our figure of merit.

- 1st Place – 3 points (least number of total bytes
- 2st Place – 2 points
- 3st Place – 1 points

Your submission must be from the code base you used

Power Efficiency

The final metric is the power efficiency of the software that you wrote. We can measure the power efficiency of your solution indirectly via the amount of current drawn from the microcontroller. Code Composer studio has an integrated tool (Energy Trace) that allows us to observe power consumed during execution.

- 1st Place – 3 points (lowest average power consumption)
- 2st Place – 2 points
- 3st Place – 1 points

Below are some resources explaining how to use EnergyTrace

Accurate EnergyTrace Measurement with MSP432

https://software-dl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_energytrace.html

https://www.youtube.com/watch?v=HqeDthLrcsg

https://energia.nu/guide/tutorials/other/tutorial_energytrace/

https://embeddedcomputing.weebly.com/ultra-low-power-with-energytrace.html

Some extra steps will be needed in order to ensure that the power measurement is fair and consistent across teams.

- First, energy trace requires that the microcontroller is plugged into the debugger, so you will be provided an extra long USB cable.
- Next, we will measure energy consumption on the simple, circular test track.
    - The code that you run for this test MUST be the same code used in the speed trial
- After you build the project, Start the debugger
- Select the energy trace icon to enable the measurement tool.
- Change the "Measurement Duration" (the clock icon in the top right hand corner of the energy trace window) to 1 minute
- While your car is on the track, select the green triangle (within the energy trace window to start the measurement, "Start Trace Collection")
- Place your robot on the track and start the program.
- The summary Energy Trace window will have your energy statistics. The figure of merit we will report is Mean Power.

Submission

Submit the following via Gradescope

A summary (via a report saved as a PDF file) of the code development explaining each of the new, individual modifications to the robot.

- The report should be detailed and each team member should write a section that explains the particular feature they were responsible for. Please make sure to indicate team roles / responsibilities somewhere in the report.
- A complete description of the code that you wrote to implement your feature. Include screenshots of your code and explain it clearly. Also include figures where appropriate (e.g. diagrams and schematics)
- A description of tests used to validate your software. Include any bugs that you had to overcome. Use the labs associated with the feature you implemented as guidelines to presenting validation results
- A video presentation of the tests you performed to validate your code. Please note, testing results should be separate from the results from the race. You must demonstrate the your feature worked prior to integration. Make sure that the results are explained, and that it is easily discernable to see that the software is functioning properly

The report should also include an extensive summary explaining the integration of code and final race results

- Include a summary of changes and modifications after the individual software modules were integrated. Tell the story of how your design changed and evolved as testing went on.
- If there are any features or additions not described in the individual reports, then insert the descriptions here.
- Videos demonstrating tests you performed to validate your robot prior to the race
    - Please make sure to include, and explain, intermediate testing results if it helps explain the evolution of your design
- An evaluation of the race results across the three metrics (speed, power, memory). Explain your results and the main contributing factors that led to each of them
- A reflection on lessons learned and suggestions for improvements of future design iterations.
- ***Please make sure that explanations are complete, but clear and concise. Include plenty of figures, diagrams and/or code excerpts where appropriate so that it is readable.***

All software written for this project should be uploaded to Gradescope in a single ZIP archive file.

- This includes the individual components and the final, integrated software.
- Please use a directory structure that is navigable and easy to understand.
- Include a README file that explains each of the folders in your submission.