Electrical, Electronic & Computer Engineering
School of Engineering & Physical Sciences

B31DG: Embedded Software 2021
Assignment 2

1. Problem

Many mobile robots use a drive mechanism known as differential drive [1-2]. It consists of 2 drive wheels mounted on a common axis, and each wheel can independently be driven either forward or back-ward. A caster wheel is usually used to passively roll along while preventing the robot from falling over.

This assignment will involve the construction of a very simplistic controller for a differential drive robot, see Figure 1.

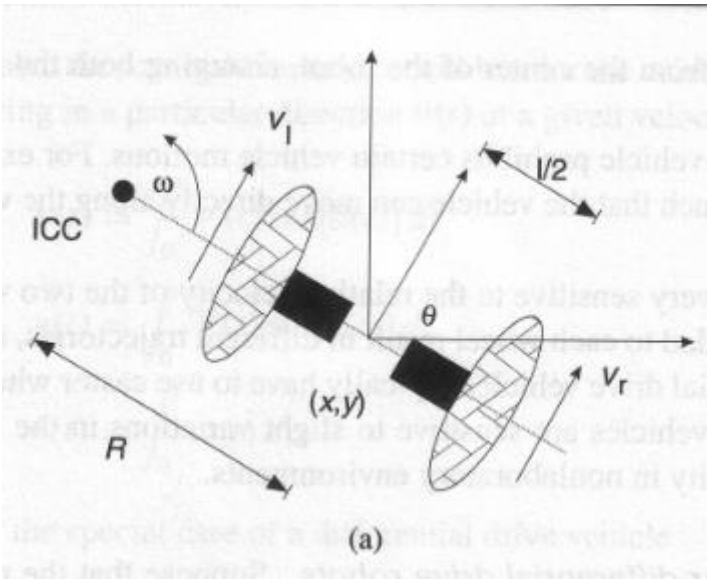

Figure 1. Differential Drive kinematics (from Dudek and Jenkin,Computational Principles of Mobile Robotics [1].

You will use the FreeRTOS for Arduino [3]. The FreeRTOS library can be installed directly through the Arduino Library Manager. For this, open Arduino IDE and go to "Sketch" > "Include Library" and click on Manage libraries. After that, type "FreeRTOS in the search window, this library will appear. After that click on the install button.

2. Simulated Hardware

The assignment is based on the sample project #45 in the category "VSM for AVR", i.e. "Arduino Motors Example" in Proteus 8 Professional, see Figure 2. This is accessible on EPS Windows servers (e.g. EM1.81 and EM2.52) through Keyserver (http://keyserver.hw.ac.uk).

The circuit consists of an ATMEGA328P and an Arduino Motor Shield [4] driving two DC motors with a L298 dual full bridge driver. The circuit is pre-wired and comes with a sample code to control the speed and the brakes of the two motors.

You are not allowed to add additional components (indeed, Proteus will not allow you to execute simulations if you add them). However, you can add instruments, such as an Oscilloscope and a Signal Generator, rewire any of the pins already used in the circuit, and save the modified sample project.

You can also re-purpose any of the GPIOs and any of the buttons provided in the sample circuit.

If you use the Arduino IDE to develop your program, a simple way to load and test it with the ATMEGA in Proteus is to:

1. Edit your Arduino IDE File/Preferences, and select the "Show verbose output during *compilation*" checkbox.
2. Export the binary of your program to your file system, by selecting "Sketch/Export compiled Binary" from the Arduino IDE menu after you have compiled your program
3. Click on the ATMEGA328P component in your Proteus circuit, and select the Program File by browsing to the location where you have stored the binary of your program.

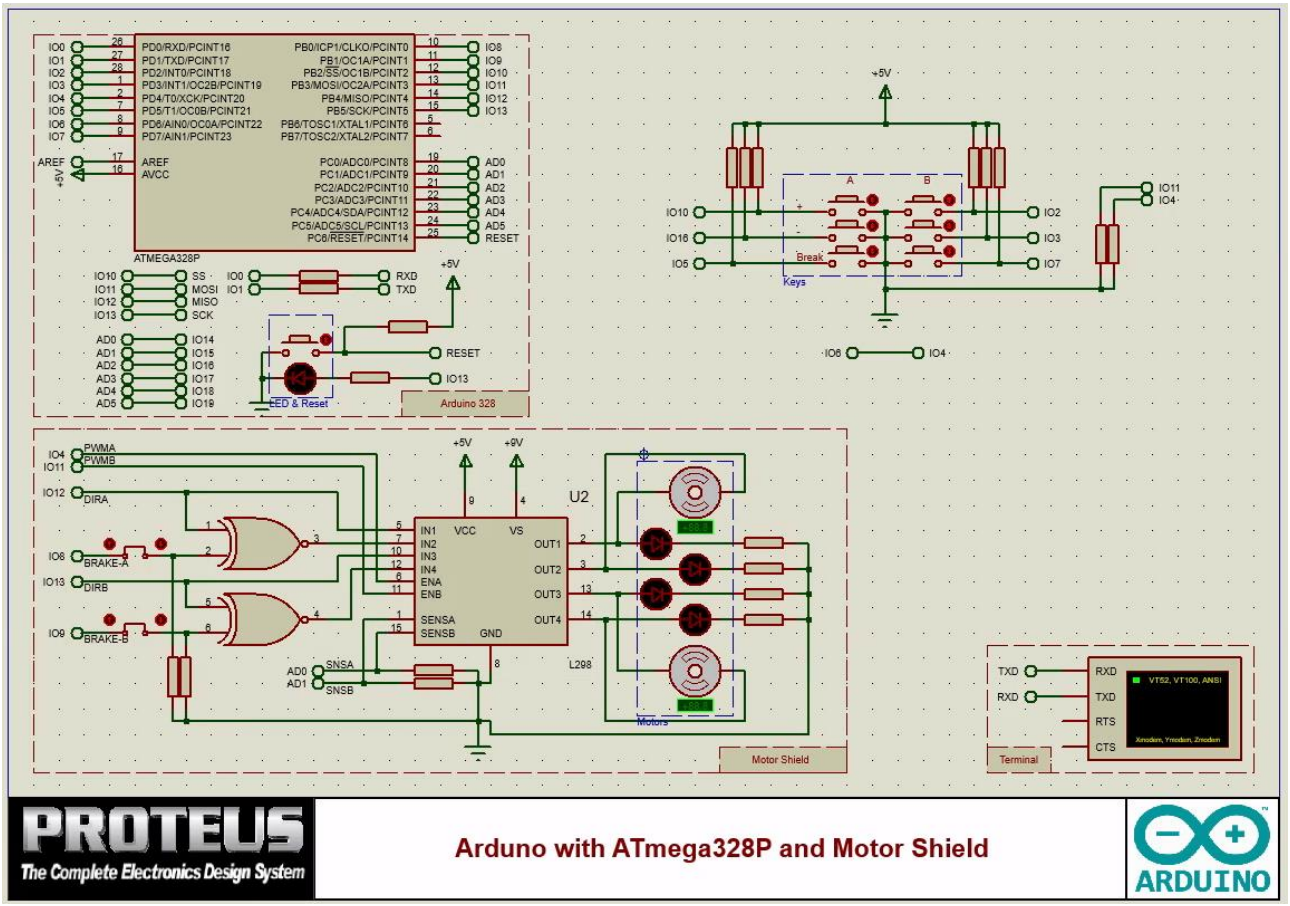

Figure 2. Proteus 8 Professional - Sample projects (*VSM for AVR \ Arduino Motors Example*). Detailed specification for each component (L298 driver, DC motors) can be accessed by clicking the components in edit mode.

3. Controller Specification

The system should provide the following features:

Table 1. Minimum required features

1	Two of the available buttons should be used to increase/decrease the input velocity for the left DC motor. The value of each increment/decrement should be 20% of the absolute maximum velocity. These keys will have no effect when the brakes are engaged.
2	Two of the available buttons should be used to increase/decrease the input velocity of the right DC motor. The value of each increment/decrement should be 20% of the absolute maximum velocity. These keys will have no effect when the brakes are engaged.
3	One of the available buttons should be used to release/engage the brakes for both motors. When brakes are engaged the velocities of both motors should be reset to 0.
4	One of the available buttons should be used to activate/de-activate a diagnostic mode.
5	When brakes are released/engaged, a message should be printed on the terminal (e.g. "BREAKES ON/OFF"~~the LED (pin 13) must be OFF/ON~~.
6	When in diagnostic mode your program should print on the terminal the maximum sensed currents for both DC motors (over the last 2 seconds and since the start of the last diagnostic mode), every 2 seconds. ~~The LED should also flash every 2 seconds.~~
7	Estimate the 2D position and the orientation of the robot, by implementing dead-reckoning using a basic forward kinematic model (e.g. assuming instantaneous acceleration, as in equations (4) and (5) in [2]). Since the motors do not have odometers, the computation should be based on the values for the PWM inputs for each of the motors. The estimated linear and rotational velocities, and the estimated position and the orientation of the robot should be printed on the terminal: (i) every time any of the velocities is updated; and (ii) every second.
8	Monitor the input velocities of both DC motors, at a 2Hz rate. When the rolling average of 6 measurements in both motors is above 88% of the maximum speed, the LED (pin 13) should flash at a rate of 4 Hz, for 3 seconds (You can ignore any other effect this will have on the motors). This feature must be suspended when in diagnostic mode.
9	Sensor processing – Your robot should carry out a periodic sensor processing task. The period should be 20ms. Since the circuit does not have sensors (other than the current sensors for the DC motors), your program will simulate the task, by keeping the CPU busy for 5ms at every cycle.

Notes
All keys must be debounced, with the only exception of when the brakes are activated, which should be triggered at the first hint that the key is being pressed.

The priority of the sensor processing task should be higher than other tasks, with the only exception of attending to the activation of the brakes. **Engaging the brakes should be considered a safety critical feature**.

4. Task 1: Design and build
Design, develop and test a C/C++ multi-task program to implement your controller.
Show it running in Proteus. The structure of your program MUST use FreeRTOS for Arduino [3].

** Build your solution incrementally, task by task. **
** Test early and test often. **
5. Task 2: Submit your solution

Include the following:

1. The source code of your program.
2. A report describing your system (5 pages max). The report should include:
    a. A description of the design of your program. This should include a dataflow diagram, and an explanation of which method(s) you have used, for instance, to implement repetitive tasks, share resources and synchronise tasks, and any assumptions you have made to design and implement your system. This should include any assumptions on the hardware of the robot, such as the diameter of the wheels and the length of the axis, and anything else that was left unspecified in the assignment.
    b. A summary of the tests you have performed on your solution, possibly including the URL to a video demonstrating your system (1 page). In addition to providing evidence that each of the required features works as specified, your tests should report the results of any measures you have carried out to assess the quality of your program. This should include the size of the binary, the size of the memory/stack, and any timing measurements you performed to show that real-time requirements are satisfied.
    c. A critical analysis of your system, discussing strengths and limitations of your design and implementation.
3. Upload a single zip file with all the above on Vision. Name the file "Mauro_Dragone_B31DG_assignment2.zip", with your name in place of mine.

6. Marking criteria

1. Work as an individual. Students MUST work on this project on their own. Please do not use code from another student or offer code to another student. Code may be run through a similarity checker.
2. The submission deadline is Wednesday 15th April.
3. Assignment is worth 35% of final mark.
4. Quality of the design and implementation is worth 25%, with full marks for:
    - All mandatory features implemented and working correctly (as documented in the report);
    - Good programming style, including good layout and commenting style;
    - Efficient, modular, easy to maintain and re-usable code;
    - Correct and effective use of FreeRTOS APIs especially to adhere to timing requirements, share resources and decouple/ synchronise tasks, while keeping the use of global variables as limited as possible, avoiding busy waits (other than for the simulated sensing processing task, where this is required).
5. Documentation is worth 10%, with full marks for:
    - Well written, well organised report, effective use of language, no typos.
    - Easy to read dataflow diagram, correct use of diagram style (among those studied in the first part of the course).

References

[1] Dudek, Gregory, and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[2] CS W4733 NOTES - Differential Drive Robots, Columbia University, Accessed 21/2/2021: http://www.cs.columbia.edu/~allen/F17/NOTES/icckinematics.pdf

[3] FreeRTOS fork of Richard Barry's freeRTOS, optimised for the Arduino AVR devices, accessed 21/2/2021, https://github.com/feilipu/Arduino_FreeRTOS_Library

[4] Alex Udanis, Arduino Motor Shield Tutorial, accessed 21/2/2021, https://www.allaboutcircuits.com/projects/arduino-motor-shield-tutorial/