Embedded Reading Task

Selections - Round III

General Instructions

Congrats on being shortlisted for the final round of the AGV embedded team selections! If you've made it so far, you definitely have made an impression on our panelists. Some of you have loads of experience with microcontrollers, analog circuits, and coding, while some of you are relatively new to it but have found a deep passion in it.

But in this round, your past experience is not what is going to be weighed. What you build here matters. We encourage you to work hard on the tasks you choose and try to show us what you are capable of.

Ask the right questions, and nature will open the doors to her secrets

Dr. C. V. Raman

Task Round Briefing

Here are some instructions that you must adhere to strictly in the duration of this round.

1. You are to attempt one of the first three tasks.
2. The fourth task (Mechatronics task on Manipulator Arm) is mandatory.
3. The reading materials associated with the chosen task are also mandatory.
4. The tasks need to be built and implemented in hardware. However, if you do not have convenient access to the required components, please refer to the alternative instructions enclosed within each task. It is also worth noting that a hardware implementation will be an enriching experience for you, and will teach you many things that simulations cannot.
5. Please do not cooperate with other students in your tasks. Plagarism in your task report submission is strictly forbidden and will be duly penalized.
6. Feel free to contact any of the people listed below, for any help or clarification that you may require.

Contacts

William E. R.	8653805579	w.ebenezer007@gmail.com
Shivam Sood	8872888172	shivamsood2000@gmail.com
Ramakrishnan C.	6381729525	krishnaram.crk@gmail.com
Sourya Roy	6290481229	souryaroy2000@gmail.com

Battery Management Task

Robotics often revolves around mobile platforms. And everything needs power. On platforms like rovers and drones, a major concern is the fulfillment of the power requirements. While battery technology has come a long way, and we can pack in a lot of energy into a single battery, we still need to be conscious of the current state of charge of the battery in order to prevent sudden loss of power and plan the duration of tasks ahead.

There are many ways to monitor the State of Charge (SoC) of a battery, and they are used in devices like laptops and mobile phones too. You too will build a system to monitor the SoC in this task.

You have to build a battery management system (BMS) for a pack of batteries (total voltage > 5 V). The BMS must not only estimate the SoC but also have reverse polarity protection, an efficient current limiting action, and offer regulated voltages via two ports offering 5 V and 3.3 V separately.

You are required to do the following:

1. Take any kind of battery you have access to. Make a battery pack with at least two to three cells in series with the total voltage not exceeding 24 V. Try to figure out a way to estimate the SOC of the battery pack, and implement it. Don't forget to add the aforementioned two ports to your circuit, for two loads to be connected.

2. On the same PCB or Breakout Board implement reverse polarity protection for the battery and make sure the current withdrawn by the load is limited to a value you deem to be safe for your arrangement of batteries. The ports must be supplying the specified regulated voltages (one with 5 V and the other 3.3 V) with all safety features incorporated.

You are expected to make a task report detailing your work in neat sections. Diagrams or figures must be clearly labelled. Comments in your codes will make it easier for us to interpret them. External references must be cited.

Optional: Implement a Kalman Filter on a your laptop interfaced with your MCU to get the SoC. You may use ROSserial.

Max. Voltage of the Battery Pack	24 V
Min. Voltage of the Battery Pack	5 V
Levels of Charge to be indicated	0% , 25%, 50%, 75%, 100%

Table 1: Specifications of the task

Judging Criteria

The SoC estimator may be implemented in various ways, broadly in two categories, which obviously will be judged on different terms.

Circuit using Analog Components:

Your understanding of the components, your skills in PCB Design, Soldering, etc..will be evaluated.

Circuit using an MCU:
Your understanding of the MCU schematic and its various functions, your skills in coding the MCU will be evaluated. You are expected to use better than normal algorithms since you now have the privilege of an MCU. Remember that high power consumption of the BMS will penalize your design!

In general: Soldering skills, robustness of circuit, innovation, compactness of PCB design, power consumption.

Resources

- A Closer Look at State of Charge (SOC) and State of Health (SOH) Estimation Techniques for Batteries (Note: Not all sections may be relevant)
- Read about MOSFETs, Tranistors, and Voltage Regulators and their working.

FAQs

1. Should we make a hardware system?
    - The hardware implementation of the concept can be achieved in various ways. It can be as simple as a simple analog circuit with some basic components or it can also be a MCU based system where some algorithms can be implemented to achieve the task.
    - In case you have access to the required components we insist you try making a circuit that can do this.
    - In case you don't have access to even basic components (like resistors, capacitors, etc..), then you are required to design a circuit with any EDA software of your choice and write a code for it. (Code is not necessary, if you are making a circuit with only analog components) Softwares like Eagle, EasyEDA, LTSpice, Proteus, TINA-TI, etc. can be used.
2. How do we demonstrate charging and discharging ?
    - Make a time-lapse video of the Indicators over the charging/discharging period. Any one would be sufficient.
3. Can we use AVR Programming?
    - Yes you can.
4. Should we simulate our circuits?
    - In case you are making a hardware system, simulation is not a necessity. But, in case you are not doing that, simulation would be an added advantage.

Motor Control Task

Part 1:
In this task, you are to upgrade a standard DC motor to a position-controlled motor (much like a servo motor). You may refer to the paper (which was also provided in your reading round) to implement a three-loop PID with different loops controlling causal variables that ultimately lead to a very precise angular position control.

You may use any alternative sensors that achieve the same results. This need not be implemented on the STM32 MCU. You may use an Arduino with an external motor driver to obtain something similar. (Hint: do not be worried about matching the exact implementation of the paper, we only need to see the three well-tuned PID loops and a precise control of the motor).

Specify the motor position using a potentiometer connected to an analog pin.

Part 2:
Now, using the highly controlled motor you developed in the previous part, make the motor move at a precise velocity, configurable by the user from outside the program. You may use a potentiometer to give this speed input.

Optional: Notice that this controller may be operated in both position control mode (Part 1) and speed control mode (Part 2). Use a simple switch to move between these modes, using the same potentiometer for both inputs.

In case you do not have access to the necessary components, you may choose to attempt the following alternative.

Implement the above task (Parts 1 and 2) to the maximum extent possible on TinkerCAD or any other simulation software (MATLAB, etc). You must also implement the mandatory task (inverse kinematics) using the setup you have built so far, for a high-precision robotic arm. Do not use prebuilt MATLAB blocks for inverse kinematics, try to implement it on your own.

Judging Criteria

Soldering skills, understanding of PID, robustness of implementation in code, innovation in state estimation, proper tuning.

For the alternate task:
Working of the TinkerCAD circuit, understanding of PID, robustness of implementation in code, effectiveness of PID tuning, proper working implementation of inverse kinematics.

Resources

- Motor Control Application Based on STM32 and PID Control Theory[1]
- MATLAB and Simulink Tutorials

[1]Paper provided only for academic purposes.

FAQs

1. Can we download readymade packages from the internet and use them?

    - You may do that for learning purposes, but nothing beyond that. Your implementation must be indigenous.

ROS Task

ROS (Robot Operating System) is a meta-operating system that is used worldwide to serve as the fundamental exoskeleton of complicated networks of sensors and pieces of codes on robotic systems. It functions as the nervous system of the robot and provides a robust platform for communication between different scripts and pieces of hardware installed on the bot.

This task is to test your understanding of the platform.

Part 1:
Make a circuit having a Joystick module which is connected to an Arduino. You must read the values of the Joystick module on both the axis and then publish these values of X and Y on 2 different topics through rosserial. Now write a node which takes these values and maps them between -255 to $+255$ from one extreme of joy to the other in a particular axis and publish these values (X and Y) to a single topic.

Part 2:
2 motors must be connected to the Arduino via a motor driver. Use of any other motor driver module (e.g. L298N) is not allowed. L293D motor driver IC must be used. The final mapped value of joy being published by you in Part 1 has to be sent back to the Arduino with the help of ROSSerial which will control both the motors (X values for motor-1 and Y values for motor-2) accordingly through PWM.

Optional: Use a game controller (with a joystick) instead of the Joystick module.

**In case you do not have access to the necessary components, you may choose to attempt the following alternative.*

Turtlesim is simple package which is popularly used to understand ROS concepts and visualize things right away.

***Part 1:**
Create a node 'turtle_driver' which uses rosparam (study about this) to take inputs from the user about the setpoint (x and y coordinates) of the turtle. Use PID to control the velocity of the turtle so as to drive the turtle to the setpoint. Make sure that the turtle is pointing in the direction it has a velocity in (no drifting or sliding turtles allowed :p).

***Part 2:**
Run another turtlenode (make sure both turtle windows are present and accounted for). You must write a node which makes one of the turtles vertically mirror the motion of the other turtle.
Beware: both the turtles are by default listening to /cmd_vel.

Optional: Implement the above alternative task on turtlebot3. Gazebo knowledge is necessary.

Judging Criteria

PCB design, soldering skills, understanding of ROS concepts, and the compactness of implementation.

For the alternate task:
Understanding of ROS concepts, compactness of implementation, problem solving, effectiveness of PID tuning, fulfillment of constraints.

Resources

- ROS Wiki - Tutorials
- ROS Wiki - Rosserial Tutorials

FAQs

1. The BMS task asked for a report. Should we make one too?
    - No. Your end product should be the soldered circuit with a rosserial interface connecting the Arduino to your laptop and producing the desired effects on the screen and in the motors. For the alternative task, only the on-screen results. We encourage you to make video recordings in case things do not work on the d-day.

Mandatory Task

Robotic manipulators are used in applications from bomb diffusal squad assisting robots to construction machinery to the massive robotic arm aboard the ISS. Most of these arms have several joints, each separately actuated. It would be an absolute nightmare to control the angles of each joint to do a simple task like grappling an object!

This is where the study of the kinematics of robots comes in. While forward kinematics tells you the exact point where the end-effector will be positioned spatially for a given set of angles at each joint, inverse kinematics tells you the angles you will need at each joint in order to move the end-effector to a particular point in space.

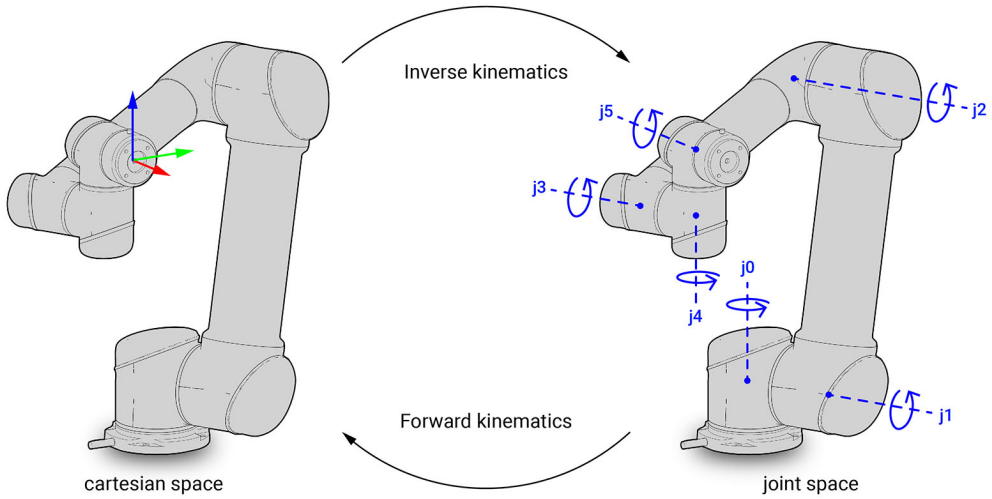

Figure 1: Kinematics of a Manipulator

Read about forward and inverse kinematics from the resources below.

Resources

- Elements of Robotics - Ben-Ari and Mondada Chapter 16 (Page 267 onwards)