Electronics and Robotics Club

BITS Pilani, K.K. Birla Goa Campus

http://erc-bpgc.github.io

Summer Assignment 2020

1 A Note from the Authors

This assignment aims to introduce you to the key areas of research in robotics through a series of stimulating tasks. It is split into three main sections - Mechanical, Electronics and Automation (coding). You can attempt whichever section that you want to explore.

Please go through the instructions carefully before attempting the questions. All the files related to the assignment questions and necessary for solving them are present in this GitHub repo.

We hope that this assignment gives you first hand experience with analysing research papers, open source culture and also with how to find solutions to problems yourself. These skills will definitely be of help in any future projects you may wish to pursue.

At first glance the questions may be intimidating to many of you. Don't panic! They are meant to be complex and challenging and will take some time and effort to complete. We want you to look around, get yourself familiar and really dig into these topics. You aren't expected to solve every question, but try as many as you can! We will be judging submission based primarily on the thought that went into the attempt - so make sure to document everything!

All the Best!

2 Logistics

2.1 Submission Guidelines

You should create a GitHub Repository consisting of all the required documents, files and code that you are required to submit for the questions you attempt. Title of the repo should be ERC-Summer-Assignment-2020. It should be properly organised with the README.md file consisting of a list of the questions you have attempted and all the files present in the repo related to that question with proper details of what each file consists of.

Attempting as much as you can of one section is enough to be considered, though you can try as many questions as you like.

All the code should be readable and properly commented at relevant places. For the tinkercad assignments, share a link to your design instead. Public repositories on GitHub are often used to share open source software. For your repository to truly be open source, you'll need to license it so that others are free to use, change, and distribute the software. Do not forget to add a license to your repository and create a separate LICENSE.md for it. A MIT license would be recommended but you can decide which one you would like using this link. You can add the license while creating the repo itself using the 'Add a License' option or you can add the license to an already existing repo using this guide.

The submission deadline is **15th June.**

2.2 Contact Information

Join our slack channel for any updates and engage in discussion [Link]. Please use the relevant channel (#general, #mech, #electronics or #automation) to ask doubts. You can also contact our team -

Atharv Sonwane (8237441175)

Vedant Shah (7359313678)

Tanmay Bhonsale (9773877028).

Mohit Chaudhari (+91 77387 55898)

Aditya Bidwai (+91 95270 51294)

Advait Kulkarni (+91 98197 55932)

Abhishek Dixit (+91 88799 23559)

3 Learning Resources and Software Prerequisites

3.0 General Note

While we have tried to include a thorough set of resources, they may end up falling short in a few places. We encourage you to search online as much as possible for any issues or doubts you may have. In some cases you may find resources better suited to you. We hope you will learn some valuable skills through this assignment and the searching for yourself will be a key part of that process.

3.1 Mechanical

1. For installing Solidworks, you can download the package for the cracked version by using proxies for torrents websites.
2. If you want to use Autodesk products, then you can download it directly from their official website using student's license (Register using BITSmail).
3. Tutorials for Simulink can be found here.
4. Lynda tutorials are the best sources for learning Solidworks from scratch (You can google it, it's easily available).
5. For visualising the linkages you can take a look at linkage software here.

3.2 Electronics

3.2.1 Tinkercad

Tinkercad is an online circuit simulator where you can implement and run circuits using Arduino. No installation is required. Get started here.

3.2.2 Arduino

The Arduino is a microcontroller that is programmed in C++ via the Arduino IDE. Since most of you might not have one, you can use the tinkercad to simulate it.

1. Jeremy Blum's Tutorials: For first time Arduino users, click here.
2. Adafruit's Learning Arduino series: If you prefer text over video, click here.

3.2.3 PCB Design

Use Autodesk Eagle to create your PCB designs.

1. Jeremy Blum's Tutorials: The first two videos are sufficient for answering the questions in this assignment.
2. Sparkfun Tutorials: If you prefer text, learn about PCB basics here and Eagle here.

3.3 Automation and Control

3.3.1 Python

This language is essential for robotics, being very easy to learn and prototype with. Check out this guide by google to start. This book and the docs are good references.

3.3.2 Linux Terminal

This will be essential for working with the various development tools you require. Get started here.

3.3.3 Git

An integral part of most software projects, Git is a tool for version control and managing changes to code. Check out this course for an intro. Make sure to sign up for the Student Developer Pack on GitHub.

3.3.4 Matplotlib

The most popular python library for plotting graphs and visualising figures. Use this series by sentdex to get started. The docs are useful too!

3.3.5 ROS

The Robotics Operating System is the backbone of every complex robotics stack. You will need to first set it up on your machine. Since ROS is supported for Linux only, we recommend that you dual boot with Ubuntu 18.04. If this isn't possible, you can try either a virtual machine or Windows Subsystem for Linux for Windows users. After getting to a Linux prompt follow the installation instructions in the wiki. Feel free to reach out to us if you run into any issues.

For introduction to the basic concepts the tutorials section of the wiki will be your best friend. A good supplement which delves a little deeper is this book by Morgan Quigley. For further resources refer to this github repo for all resources to learn ROS.

3.3.6 Turtlebot + Gazebo

Gazebo is one of the most popular simulations for robotics with Turtlebot being a versatile platform within it. Have a look at this video to know how to set up Turtlebot in the Gazebo simulator. You can also have a look at this youtube playlist for knowledge of handling URDFs and working with Gazebo using ROS.

3.3.7 OpenCV

OpenCV is a computer vision library which can be used for processing and feature extraction on images. You can try out the tutorials for the library or even the sentdex videos series to get some hands on experience.

3.3.8 ML Tools + Frameworks

ML is pretty complex, these libraries make things easier. Numpy can be used for basic linear algebra. PyTorch and Keras are the two most popular libraries for creating Deep Learning Models.

3.3.9 Deep Learning

DL is a complex and broad field. CS231n is a great introductory course. For a more thorough treatment, refer to this book by Ian Goodfellow.

3.3.10 Reinforcement Learning

For a very high overview of the field check out this video series by Brian Douglas. For a more in-depth intro you can try the classic lecture series by David Silver or Stanford's CS234 for a more recent course. This book by Sutton and Barto is https://spinningup.openai.com/en/latest/index.htmlalso a great reference. For help with algorithm implementations check out Spinning Up by OpenAI

3.3.11 Motion Planning for Robotics

Use these (1, 2) books for a brief introduction and reference.

3.3.11 Control Theory

Robotics deals with a lot of continuously operating, dynamic systems which are dealt with using tools of Control Theory. For the basic concepts check out video series by Brian Douglas (1, 2) and Steve Brunton (1).

4 Assignment

4.1 Mechanics

4.1.1 Simulation:

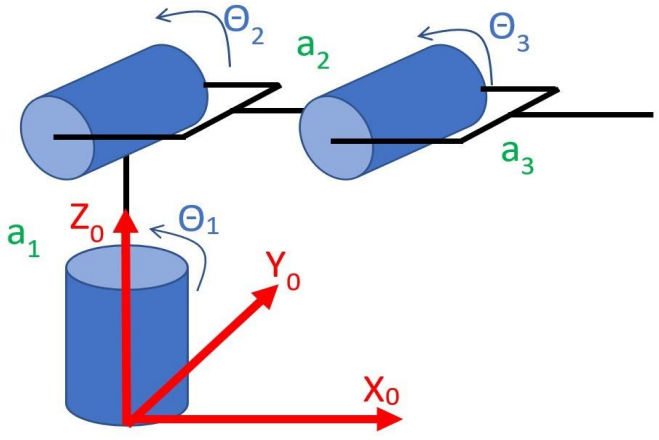
Simulate the following linkages on Simulink:

1. Watt II linkage
2. Stephenson II Linkage
3. Stephenson III Linkage

4.1.2 Designing:

Design an isometric model of a chess bot with justifications to your design on Solidworks or AutoCAD.

4.1.3 Inverse Kinematics:



This is the kinematic diagram of an articulated manipulator. Find the inverse kinematic equations. Consider the following 2 cases ($a_1 = 10cm, a_2 = 7cm, a_3 = 5cm$).

1. We want the end effector to be located at position 7cm in X_0 direction, 4cm in Y_0 direction and 7cm in Z_0 direction.
2. We want the end effector to be located at position 3cm in X_0 direction, 5.25cm in Y_0 direction and 8.5cm in Z_0 direction.

   Find the values of θ_1, θ_2 and θ_3 in all the cases.

4.1.4 Research Paper Analysis:

Go through this research paper and discuss the given linkages and their performance against a) parabolic function b) range ballistic function and c) elevated ballistic function. Try to replicate the graphs given in the paper.

4.1.5 Mathematical method to overcome limitations in a robotic arm

Differential equations (DEs) are an integral part of overcoming the problems encountered in designing a robot. Since analytical solutions to these DEs are not always obtainable or impractical, a numerical approach to them forms a crucial part in solving them.

Go through the following paper and try to implement the same results using Runge-Kutta sixth order six stage numerical techniques. You should expect to get the same results. Also, graphs for the data calculated have to be plotted on Google colab / MATLAB for both the algorithms. Solve it for 3 different sets of constants in both RK(5, 5) and RK(6, 6).

4.2 Electronics:

4.2.1 Arduino

Simulate the following circuits on tinkercad's circuit simulator. Remember to comment your code well and keep wiring neat and tidy.

1. Use an arduino to measure the value of a resistor (between 0 and 1K), and display the value on an LCD screen.
2. Use a potentiometer to control angle of a servo motor and display the current angle on a 3-digit 7-segment display using an arduino as microcontroller. (Hint: You can use the CD4511 IC).
3. Create a circuit using Arduino and an IR sensor/remote. The user should be able to use the remote to do the following:
    a. Control the brightness of a light bulb (not LED).
    b. Set the speed of a DC motor as high, medium, low, or off.
4. Create a circuit consisting of 2 arduinos. One arduino asks the user to enter a single line message, followed by a 4 digit pin, through the serial monitor. Then it encrypts the message and sends it to the 2nd arduino using I2C. The 2nd arduino then prompts the user to enter a pin via a keypad. If the pin is correct, it displays the decrypted message on an LCD, and if it is incorrect, it sounds an alarm using a piezo speaker.

4.2.2 Digital and Analog Circuits

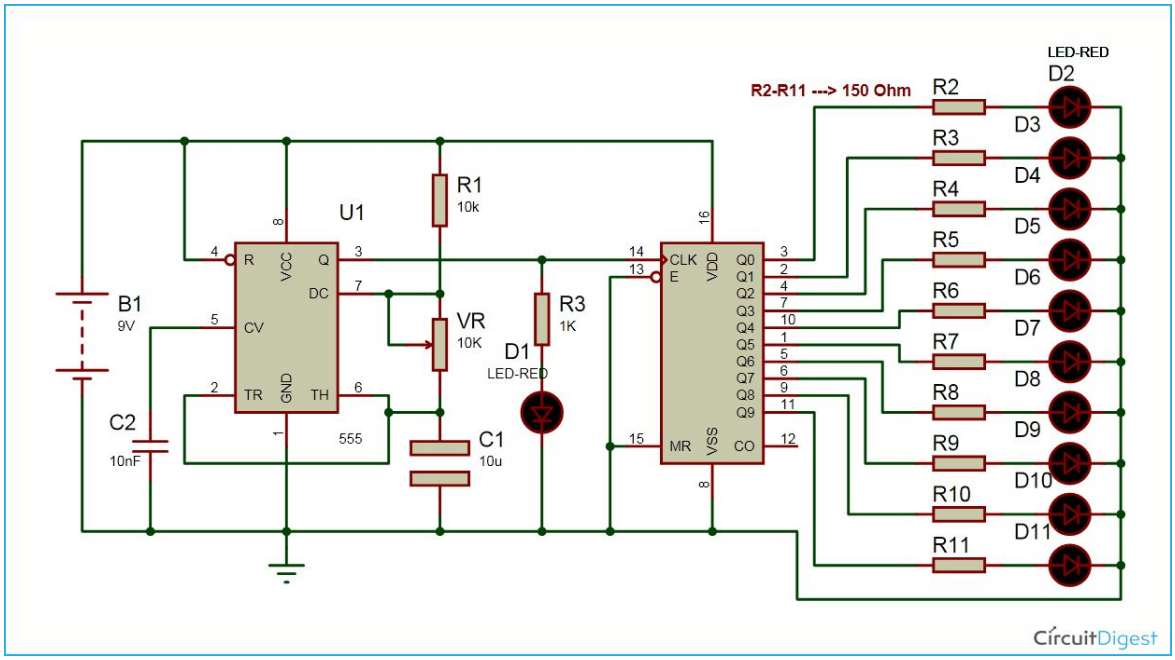
Implement the following circuits in tinkercad, without using an Arduino or microcontroller:

1. Design a circuit for an automatic LED night light using LDR and LM741, which is an op-amp IC.
2. Study the datasheets of the 555, 4511 and 74HC93 ICs. Use them to create a circuit that counts from 0 to 99 on a 2-digit 7-segment display.

4.2.3 PCB Design

Use Eagle to design the following PCBs. Use no more than two copper layers. When submitting your solutions, include both board (.brd) and schematic (.sch) files.

1. Design the most compact possible PCB for the following schematic, as seen here. Bonus points on designing a silkscreen.



2. Design a PCB for the circuit you designed in 3.2.2 (2), i.e. 2-digit counter using 555, 4511 and 74HC93.

4.3 Automation and Control

For every question that you attempt from this section, you should submit well documented code along with detailed instructions on how to reproduce your results. For questions involving ROS, upload the src folder of your workspaces to the submission repository along with a rosbag recording.

Each question (apart from the Paper Review) has two options which you can attempt.

4.3.1 Basic Robotics

Attempt either one of the following. We recommend you attempt the first option if you are not familiar with ROS.

1. ROS & Basic Control

   This question has two compulsory parts.

   A. **ROS** is an open-source operating system for your Robot. It provides all the functionalities of a standard operating system such as control, hardware interfacing etc. ROS consists of a graph-like framework consisting of code segments (known as nodes) - each with a particular purpose interconnected by message channels (known as topics) which are used to interchange messages between the nodes. For more information on ROS and how to set it up on your machine, refer to the *Learning Resources* Section.

   Your task is to write a ROS publisher node to such to /cmd_vel topic such that the Turtlebot will move in a circle in Gazebo. Gazebo is a simulation environment which can be integrated with ROS. Turtlebot is a specific model of robot available for simulation within Gazebo. Along with the publisher code you also need to submit a screenshot of the graph framework which can be generated using the rqt_graph command and a rosbag recording of the functioning robot.

B. PID (**proportional–integral–derivative**) is a widely used technique in robotics control. For more understanding on a PID controller check out this video series or read the wikipedia page. Your task is to write a python script to implement a PID controller. Along with this, try to give a visual representation for the same using plots (matplotlib is recommended).

2. Omnibase

   Omnibase is an **omni-wheel based ground robot** platform for simulation built by ERC members. Your task is to use it to make a simple waypoint controller with PID to traverse a given path in Gazebo. Given a list of points, make a ROS node which subscribes to /odom topic (for current position) and publishes to /cmd_vel topic the required velocity to get to the next point. The robot should stop at each point in the path before continuing. It will receive the path by subscribing to /path topic with /nav_msgs/Path message type. For the purpose of demonstration you can set up a simple node for publishing /path.

   Since the robot uses omni-wheels it can move in any direction without needing to rotate. Your controller should use this fact to publish velocity in x and y direction [Hint: you might need to implement PID separately for x and y]. Submit your code and a rosbag recording of the functioning robot.

4.3.2 Research Paper Review

Path planning is a fundamental problem within robotics automation. Since many robotics use-cases involve unknown and dynamic environments, a common approach to design a planner is to use probabilistic methods to sample from the known area and then computing a path. This paper discusses three different such algorithms RRT, RRT* and RRG. Another probabilistic algorithm is PRM (Probabilistic Road Map).

Your task is to choose one of these four algorithms and implement it. The end result should be a demonstration of the algorithm planning a path through a field

of obstacles. We recommend that you use matplotlib for visualisation of the obstacles and the path. You can take obstacles as simple oblique polygons.

Along with the code and the demonstration plots, you should also submit a short (1-page) summary of the conclusions in the paper on the differences between the algorithms and also differences between RRT based methods and PRM.

4.3.3 ML in Robotics

Attempt either one of the following.

1. Computer Vision

   CV involves use of various techniques to derive meaning out of images. This is especially important for robot perception where the robot should identify and analyse its environment. OpenCV is a library focused on providing real time vision capabilities. The first part of your task is to get familiar with the library (available in both python and C++) and to use it for creating a shape recognition program. When given an image (example) containing different shapes of different colours, the program should output a list of the shapes and what colour they are.

   Similar techniques are extensively used in real life applications like self driving cars. The second part of the task is lane detection - given a video from a front facing camera of a car (like the one in the assignment repo mentioned in section 1), detect and annotate lane marking using OpenCV.

2. Basic Reinforcement Learning

   RL is a machine learning technique in which agents are trained to take actions in an environment with the objective of maximising a reward. It has a large potential in the field of robotics controls and automation. Your task is to create an agent to solve the Cartpole-v0 environment in OpenAIs gym. OpenAI gym is a collection of various environments and games on which to test RL algorithms. In the Cartpole environment, the

agent has to give left or right pushes to an inverse pendulum to keep it upright.

You can use Q-table learning (easier), Deep Q Learning (harder) or any other techniques you wish. The aim of this question is to get you familiar with the gym environment and the general structure of RL agents. We recommend using Google Colab (a jupyter notebook based online development platform) for this task. You can either submit a notebook or python files with thorough instructions on how to reproduce your results.

4.3.4 Advanced Robotics:

Attempt either one of the following.

1. Computer Vision + ROS

   Create a **line following bot** which follows a line using computer vision. You may use the available turtlebot platform in ROS. The line following bot must follow the line in the world file provided to you (in the repo mentioned in section 1). You will have to use OpenCV to detect the line, its centre and thereby alter the velocities of the bot to minimize the error between the bot's centre and the line's centre. Submit your code and a rosbag recording of the functioning robot.

2. Reinforcement Learning + Control

   LQR (**linear-quadratic-regulator)** is a control technique widely used to create an optimal controller dynamic system with quadratic cost functions. This video series provides a good introduction.

   In this task, you will be using Reinforcement Learning to create a simple LQR controller. This is one of the first papers to use this technique. In the paper, they used the Recursive Least Squares method to fit the policy. Your task is to use neural networks to represent the policy which takes current state and outputs the required control signal. To train this neural network you can use Vanilla Policy Gradient or if you are comfortable with

Deep RL you could try PPO or DDPG. Similar work has been done in this paper which combined Deep RL and LQR.

You must demonstrate the controller through a simulation of the Cartpole-v0 task in OpenAI gym. If you want more of a challenge, you could also try the Acrobot-v1 task. Along with the code and the demonstration, you should also submit a 1 page summary of the differences you found between the old and new paper.