# RECRUITMENT SCREENING APPLICATION

## abhiyaan-electronics-module.brd

## Instructions

- Try to attempt as many questions as possible. Please do note that even if you can't get a question right, do explain your approach. It will help us understand your thought process which is what we are looking for.
- Add all the files and codes you used to derive your answer in your final submission. Remember to mention all the references and resources you used to arrive at your answer.
- For any queries and clarifications contact:
  - Vamsi Krishna (EE21B153)(99529 12710) - Question 1
  - Kaushik M. R. (CH21B044)(93644 72454) - Question 2
  - Dhaksin Prabu (EE21B039)(80724 62970) - Question 3
  - Niranjan Kartha (EE21B095)(81293 94578) - Question 4
  - Kesava Aruna Prakash (EE20B061)(97910 99381) - Question 5

- Assume the person reading your answers is a layman and try to explain your answers accordingly wherever you can.
- Link for submission of your answers. Also add a short intro in the format shown in the next page.

# ALL THE BEST!!!

Submission for Team Abhiyaan
==========

(Example)

**Name:**
Dexter Morgan

**Roll no:**
XX2YB069

**Previous Experience**
10 Years of Experience in handling dangerous tools

**Current PORs:**
Forensic Technician for Miami Police Department

**Why I want to work in the team:**
Because I think the team is the best

**Relevant Courses:**
*In Institute*
CS110
0
ME220
W
NA525
1

0 - Passed
W - Attendance Shortage
1 - Doing

*Online*

CS20SI
Andrew NG Deeplearning.ai
David Silver RL

Other Relevant Things:

Introvert
Committed to the work

# QUESTION 1

- **Rotary Encoders:**
  Any form of speed or position control would require feedback about the system's current angular speed or position. Rotary Encoders are electromechanical sensors that generate output signals based on the shaft's motion, which can be interpreted to give the desired angular position or velocity.

  Both Vikram and Bolt use several encoders. Read up on Incremental Quadrature Encoders (https://en.wikipedia.org/wiki/Incremental_encoder) and Absolute Encoders (https://en.wikipedia.org/wiki/Rotary_encoder#Absolute ).

  In the following sub-questions, we shall explore how we use encoders within our electrical systems

- **Incremental Encoders:**
  All our motors are connected to an incremental encoder each to generate feedback on the speed of the motor shaft. The feedback pulses of the encoders are processed by microcontrollers, which publish this data to the onboard computer via USB - Serial connection. We want you to demonstrate a part of the implementation below.

  Use the simulator at https://wokwi.com/ to create a circuit involving only an Arduino Uno and an incremental encoder. Read the phase A and B inputs through the required pins.

**a) Devise a way to input the phase A and B signals and count the number of ticks in a fixed time interval (T) of your choice. Publish this count through the Serial Monitor after every T units of time. Explain your implementation in brief.**
*(Hint: Look up "Interrupts" to handle these pulses efficiently)*

**b) Derive an expression that uses this tick count and time interval T to give the Rotations per Minute (rpm) of the motor shaft the encoder is attached to. Mention any other variables required and/or suitable assumptions that you make in the process.**

- **Absolute Encoders:**
  Our Steering system feedback on our autonomous golf cart relies on a 10-bit resolution absolute encoder to generate positional feedback.

**Assume you have a similar encoder with 10 output bits ($2^0$ to $2^9$), which gives an output 0 or 1 compatible with Arduino Uno logic levels. Describe how you would use the Arduino UNO to find the value of the absolute position (Between 0 and 1023).**

**For bonus points, you could simulate the code on some simulation platform like https://tinkercad.com or https://wokwi.com/ by using toggle switches to represent a 0 or 1 data level and input it to an Arduino UNO and calculate the position using your above implementation.**

It turns out that to map the angle of the steering wheel completely, we require around 3 complete revolutions of the encoder. **Write a pseudo code to use the same single-turn encoder and expand it to any number of turns. The resultant output should span at least 1024*3, for example, -1024 to 2048. You can assume that the steering always starts from the center - Value at the center given by the steering encoder: 512.**

c) **A dataset attached contains a sample set of encoder pin readings taken from our steering system. The first line gives the corresponding power of 2 the bit must be raised to. Write a Python or C/C++ program that can process this data and use your logic for the above two questions to generate an output value reading. You can make any assumptions but state them clearly.**

- **Fault Detection:**
  Wiring faults can cause significant damage since we get incorrect feedback. We need to develop a live diagnostic tool that runs in the background and verifies if the output is sensible.

d) **You are given a few sample datasets of raw readings from an absolute encoder that are caused by faulty wiring(loose connection & incorrect wiring). Try to identify the fault in each case and explain how you found it. Determine some logic that can identify these faults during runtime and write a pseudocode.**
*(Note: Assume faults are only in some of the higher-order bits). State any assumptions you make clearly.*

**Bonus points if you write a code that takes this data as input and determines which pins are responsible for the above faults.**

*We use the following encoders in our systems if you're interested.*
*Incremental*
*Absolute*

## QUESTION-2

In most systems, we know what output we desire, but not the input we need to make it happen. Let's say you drive a vehicle. You want to run the vehicle at 20 km/hr, but now you want to accelerate to 30 km/hr. You increase your throttle until you reach your desired speed. Throughout you are checking your "error", meaning how "far" your speed is from your desired speed, and increase it until it is close to zero. This is called "feedback" and is given by encoders and other sensors on the vehicle.

A Control System provides the desired output response by controlling the input.

Control systems are classified as Open and Closed Loop systems. The presence of a feedback path is the primary difference between these two systems.

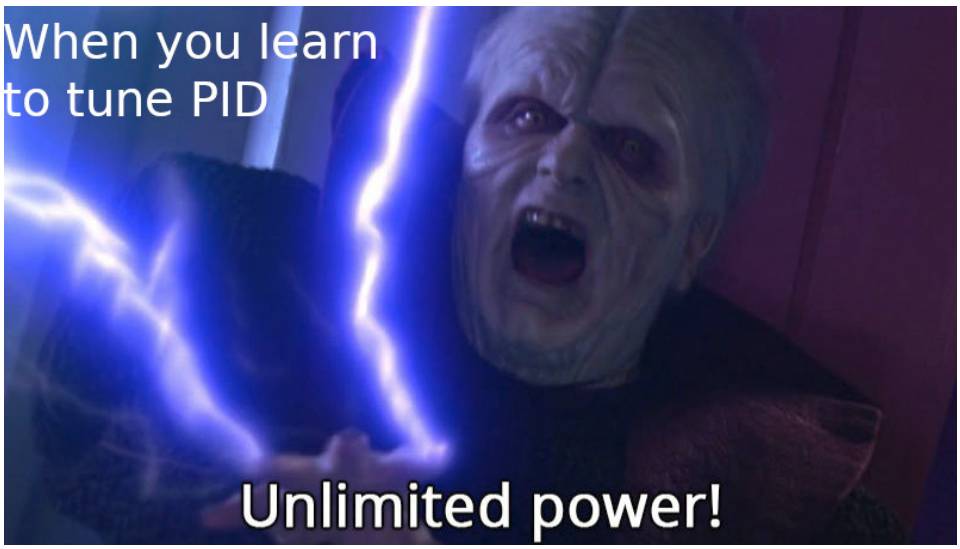
**a) Explain the working of Open and Closed Loop Control systems with another example for each.**

**b) You are working in a huge automotive company and tasked with designing a control system to control the motor speed. What are the parameters that will determine whether the designed system is industrially good?**

For a closed loop system, the "path" taken to attain the desired setpoint is crucial. In our earlier example, you could either slowly increase the speed, which might take very long to reach, or you could randomly try increasing the throttle to some higher values, and either reach there faster or crash the vehicle which is probably not optimal.

- PID Control Systems:

  PID controllers are found in a wide range of applications for industrial process control. Approximately 95% of the closed-loop operations of the industrial automation sector use PID controllers. PID stands for Proportional-Integral-Derivative. These three parts are combined such that it produces a smooth control signal. As a feedback controller, it delivers the control output at desired levels.




**c) Read more about PID controllers and explain in brief how each of these parts is significant.**

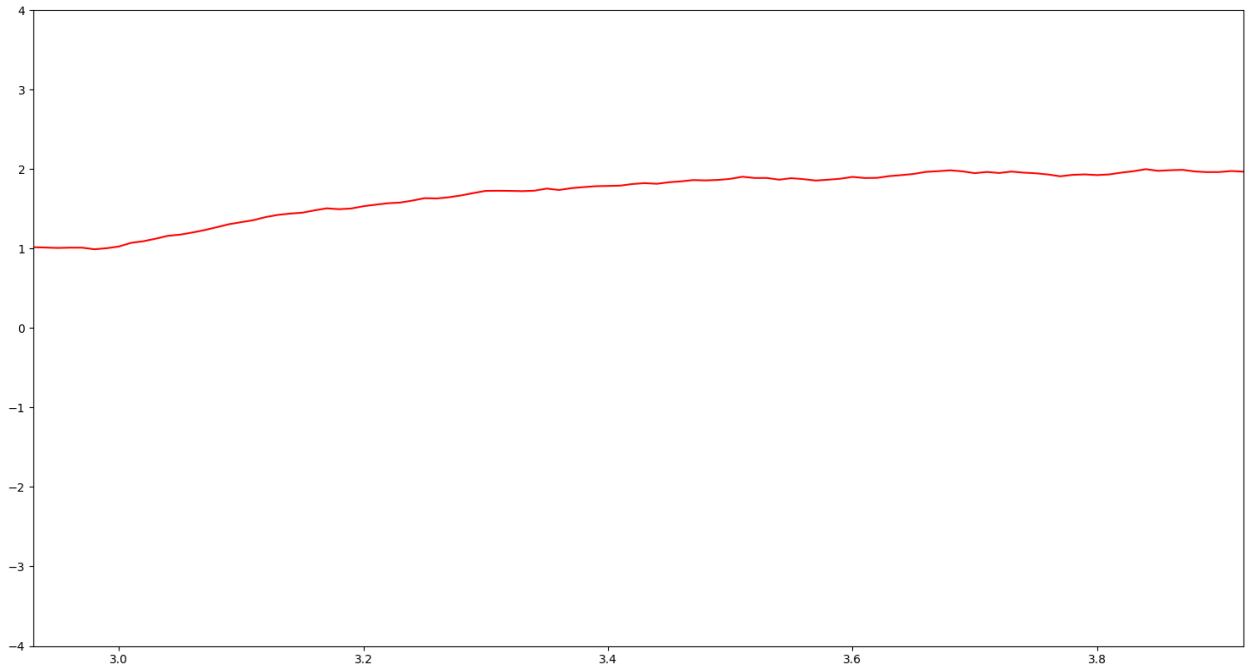
It is not always optimal to test whatever control system you design directly on the vehicle, hence it is preferred to model your system first and simulate its behavior.

A python script is attached which has an idealistic model for Bolt, with the equation

$$\frac{dv}{dt} = k_1 \times throttle - k_2 \times v + noise$$

It also has a PID controller with a graph that you can use to visualize the changes.

An example is shown below, where the vehicle moving with a velocity of 1m/s until t = 3s at throttle = 20 is accelerated with throttle = 40 to saturate at velocity 2m/s.




The dataset of the variation of v with time is attached.

**d) Write a code that uses this dataset to optimize and find the values of k1 and k2. You might have to solve the differential equation first. (Python libraries like scipy.optimize.curve_fit ,etc can be used for this purpose)**

If you aren't able to do the above, you can plot the curve equation on Desmos or any other graphing calculator and change k1 and k2 until you get an approximately similar curve to proceed with the next part of the question.

Now open the python script and look for the line

```
bolt = VelocityModel(k1=1, k2=1)
```

Replace k1 and k2 with the numbers you found, now look for the line

```
controller = PIDController(bolt, kp=1, ki=1, kd=1)
```

Set ***kp***, ***ki***, and ***kd*** with any initial values and run the code.
(Note you would have to install some libraries if you haven't already)

The code will keep updating the setpoint velocity and you can visualize how your system behaves.

e) **Comment on the graph. Tune your model by finding *kp*, *ki*, and *kd* that make it smoothly and quickly reach the setpoint, without overshooting or oscillating around it. Understand how *kp*, *ki*, and *kd* affect the control. Explain your thought process along the way whenever you change these constants.**

# QUESTION-3

## CONTROLLER AREA NETWORK:

Controller Area Network popularly known as CAN is an important communication protocol used in nearly every vehicle you see today ranging from Tata Nano to Tesla X series.

- What is CAN?

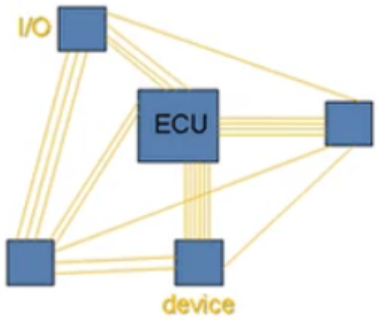
Ever heard of **kaiten-zushi** styled restaurants?

Kaiten-zushi is a type of Japanese restaurant that serves sushi dishes using a conveyor belt. In a kaiten-zushi restaurant, sushi chefs prepare various sushi dishes and place them on small plates or bowls, which are then placed on a conveyor belt that travels around the restaurant, passing by each table. Customers can simply pick up the dishes they want

as they pass by, and pay for their meal based on the number and type of dishes they have taken.

In a kaiten-zushi restaurant, the conveyor belt carries the sushi dishes around the restaurant. Similar to Kaiten-zushi, nodes in the CAN network(chefs and customers), can publish and receive messages(food) from the bus(conveyor belt)

**a) You now have some idea about how CAN works. Now explain the disadvantages and advantages of CAN instead of the communication network shown below.**




**b) CAN protocol uses 2 data lines to send information across namely CAN-H and CAN-L. What is the significance of using 2 lines? Explain the advantages and disadvantages of this method.**

Another similarity is that both kaiten-zushi and CAN rely on a standardized protocol to ensure smooth operation. In a kaiten-zushi restaurant, the dishes are labeled with a standardized color-coding system that indicates their price. In a CAN network, messages are transmitted along with a message ID that specifies the format and priority of each message.

**We will be using Extended CAN frames for all the questions asked below.**

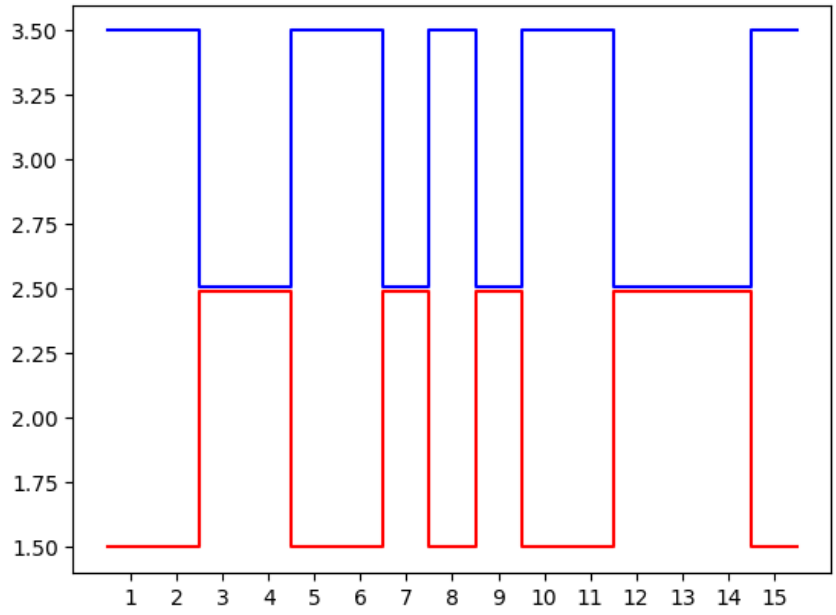

| Field name | Length (bits) | Purpose |
|---|---|---|
| Start-of-frame | 1 | Denotes the start of frame transmission |
| Identifier A (green) | 11 | First part of the (unique) identifier which also represents the message priority |
| Substitute remote request (SRR) | 1 | Must be recessive (1) |
| Identifier extension bit (IDE) | 1 | Must be recessive (1) for extended frame format with 29-bit identifiers |
| Identifier B (green) | 18 | Second part of the (unique) identifier which also represents the message priority |
| Remote transmission request (RTR) (blue) | 1 | Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below) |
| Reserved bits (r1, r0) | 2 | Reserved bits which must be set dominant (0), but accepted as either dominant or recessive |
| Data length code (DLC) (yellow) | 4 | Number of bytes of data (0–8 bytes)[a] |
| Data field (red) | 0–64 (0-8 bytes) | Data to be transmitted (length dictated by DLC field) |
| CRC | 15 | Cyclic redundancy check |
| CRC delimiter | 1 | Must be recessive (1) |
| ACK slot | 1 | Transmitter sends recessive (1) and any receiver can assert a dominant (0) |
| ACK delimiter | 1 | Must be recessive (1) |
| End-of-frame (EOF) | 7 | Must be recessive (1) |

**Understand what each field(set of bits : SOF, ID, SRR, IDE etc. represents) does** to answer the questions below.

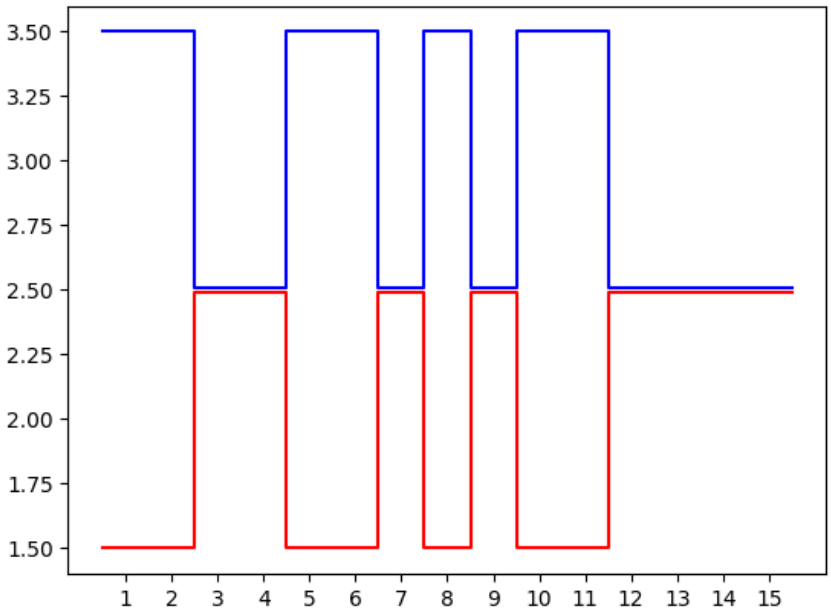
When 2 CAN messages are pushed into the bus, the messages are sent in the bus based on the priority given in the CAN frame.

**c) Given below is the first 15 bits of a CAN frame observed using an oscilloscope(more like simulated using python).Decode the oscilloscope data and find the priority order of all the frames.**

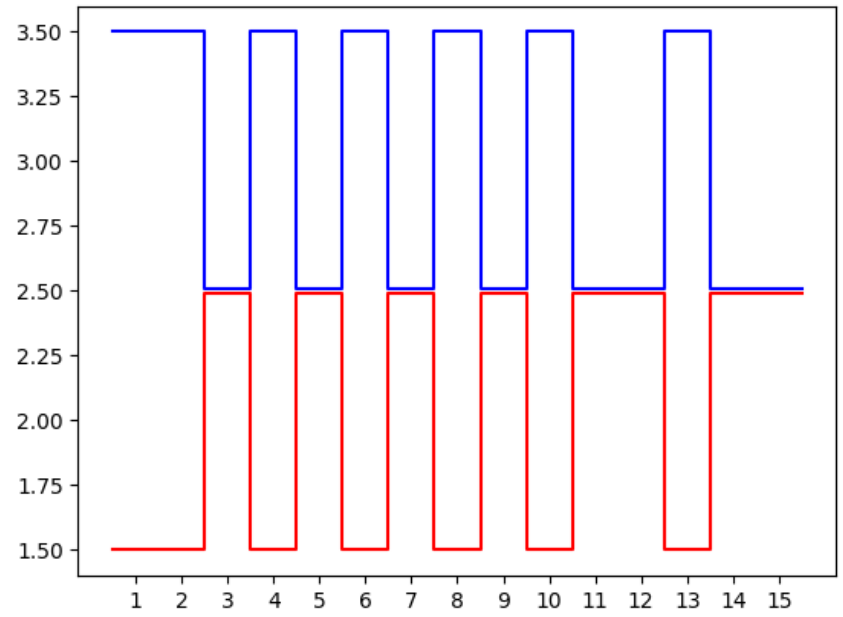
i)



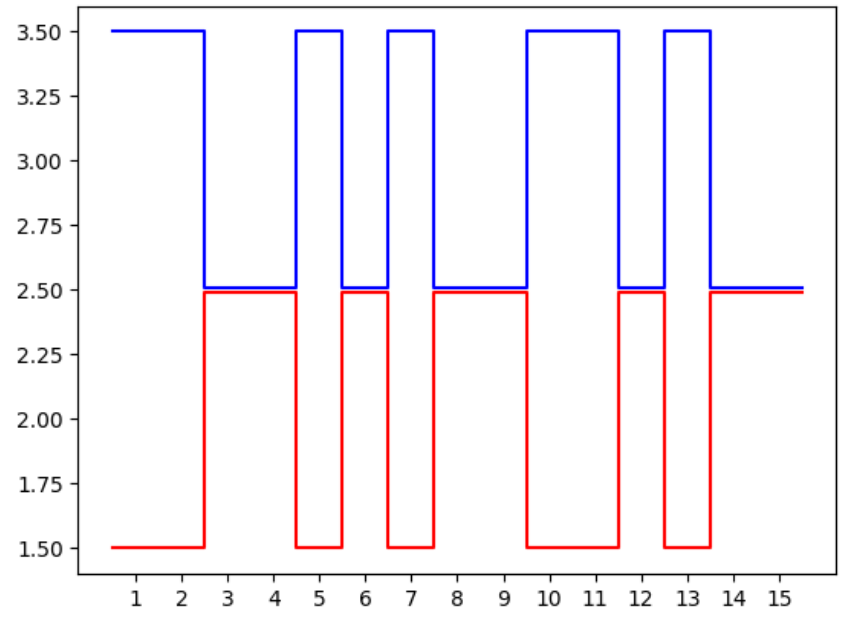

ii)



iii)


3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

iv)


3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

**d) Given below are extended CAN frames in Hexadecimal format. Find the priority order and the data present in each message and explain your process.**

**i) 0x127D555503876B92FF – 9 bytes**
**ii) 0x127EAAAA046798B912FF – 10 bytes**
**iii) 0x27EAAAA059867F922FF – 10 bytes**

**e) Prioritization of electronic modules inside a vehicle is one of the most important safety factors.**

**a. Brakes**
**b. Steering**
**c. Throttle**
**d. Emergency Stop**
**e. Head Lights**
**f. Brake Lights**
**g. Horn**
**h. longitudinal velocity feedback**
**i. Steering angle feedback**
**j. Door lock/unlock**
**k. Battery Monitoring details**

**Assign a message ID for each of the modules mentioned above and justify your priority order.**

## Filtering and Masking

When a particular dish is available on the conveyor belt, a person has to choose whether to take the food or not. Similarly each node in the CAN bus can decide whether a message has to be ignored or not.

**f) Explain the algorithm used by each node to determine whether a message has to be ignored or not (Hint: Masking & Filtering)**

**g) Find the number of IDs the node accepts for the given mask and filter**

a) Mask = 0x15D75D7D, Filter = 0x13579BDF
b) Mask = 0x12D5AFAA, Filter = 0xFDB97531

## Error Handling

One of the known features of CAN communication protocol is a relatively better error handling than other communication systems.

Standard CAN CRC polynomial is shown below

$$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$$

**h) Explain Acknowledgement error and Cyclic Redundancy Error briefly.**

**i) When a Transmitter sends a CAN frame 0x493FFC33021E9FEBFF what happens to the frame when an acknowledgement error happens and what happens to the framewhen a receiver acknowledges the message?**

**j) A transmitter sends a CAN frame of 0x527EAAAA0618B3200003FF, and the receiver reads a CAN frame of 0x527EAAAA063166400002FF. Does this result in Cyclic Redundancy Error or not? justify your answer.The probability of a bit flip is 1% in the data field, what is the probability that this particular event happens?**

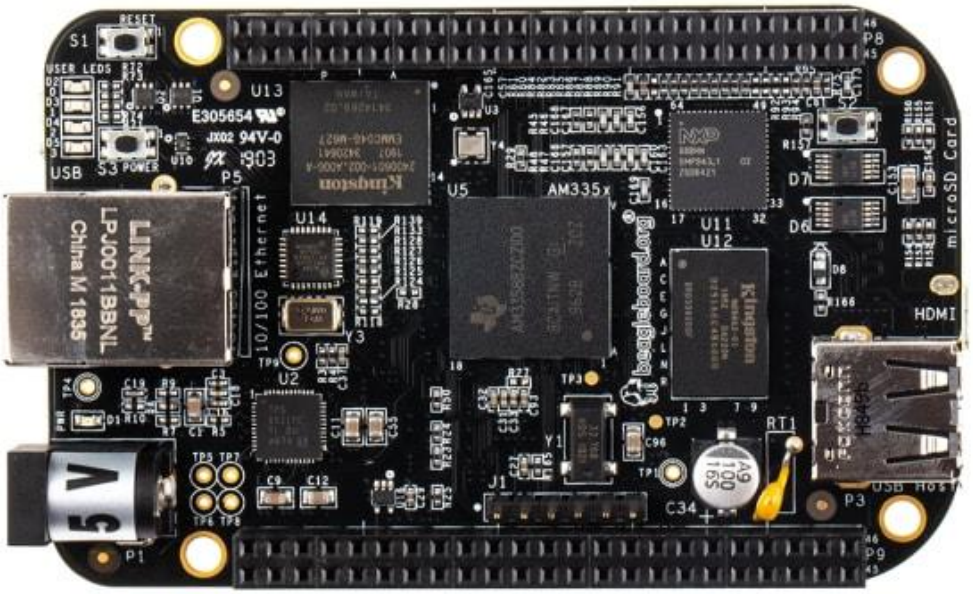
## REFERENCES


- CAN
- Cyclic Redundancy Check
- Hexadecimal to binary (will be needed for conversion of large bytes of data)

# QUESTION-4

## 1. Linux

We use something called a “single-board computer” (SBC) as the brains of our electronics system. It’s a computer like your laptop but *way* smaller – it’s literally just one small PCB. You can see it as the little brother of the massive one that the Software module uses.




BeagleBone, an example of an SBC. This will fit in the palm of your hand.

We use an SBC to coordinate information between Big Brother and the microcontrollers that control the motors in our system. Our SBC, currently a Raspberry Pi, runs on Linux (specifically, Ubuntu Server 20.04). So it’s very important to get comfortable with Linux.

## 1.1 Install Linux onto a VM

It's kinda scary to get into Linux and use the command line the first time. So to make things easier, you could practice on a Virtual Machine (VM). You can think of a VM like a computer inside a computer – you use your laptop (the "host") to run another operating system like Linux inside it (the "guest").

If you're on Windows or macOS, you need to install special software (called a "hypervisor") to create a VM. Your first job is to install one of these. On Windows, common hypervisors include VirtualBox and VMware. macOS has UTM.

**Your first task is to install a Hypervisor, and then install Ubuntu 20.04 on it. *The version of Ubuntu is important*.Add screenshots, document your process. Mention any online guides that you may have found and used.**

**Bonus:** (optional – do this if you're done with everything else and have nothing better to do): Running Linux on a VM is what some people in the Linux community may refer to as "cringe". Install Linux on hardware instead of on a VM. Shrink your partition and install Ubuntu 20.04 alongside your main OS.

Come on. Install Linux. Embrace free software. Make RMS proud.

**Extra extra bonus:** (optional (optional) x2): If you already have experience with Linux, run a docker image for Ubuntu 20.04, sharing a common folder `~/abhiyaan` between your container and host, and continue with the rest of the questions in that container. Also make sure that the container has access to all the USB devices running on your host, and set up all the extra permissions you may need to run ROS. If you do this, I will buy you a Fresh Lime (or a Chai if your throat is messed up like the rest of us).

## 1.2 Install some packages

On Windows, when you want to download some software, you're probably familiar with the process of going to that software's website, downloading an installer, and then running it. On Linux, we use the "package manager" through the command line. So we can install anything we want without having to leave the comfort of our terminal!

**Your next task is to install `python3` and `git` through Ubuntu's package manager. Look up the commands you need, open up a terminal, and run them. Next, install the `crc` package using Python's package manager.**

**Also, what other ways exist to install packages on Ubuntu? What are the advantages/disadvantages of those methods? Don't forget to mention the guides you used, and add screenshots!**

## 1.3 Using the terminal

The command line is a very powerful tool. If you properly learn how to use it, you can do very creative things with your computer to make your life much easier. It also makes you look very cool while using it.

This is what others see you as when you open a terminal

Here's a nice introductory guide.

**Part A – Perform the following tasks using shell commands:**

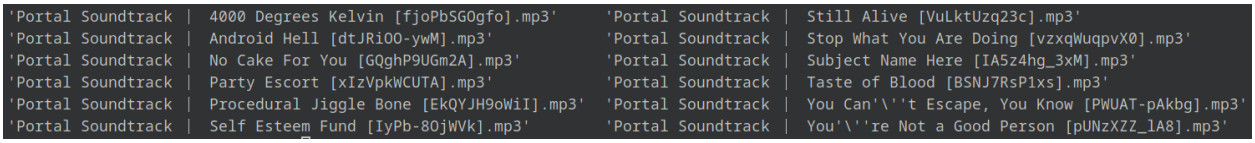
- Make a folder called abhiyaan in your home directory.
- Change your working directory to abhiyaan.
- Make a file called hello.txt.
- Open hello.txt with a text editor (like nano) and write a few lines of text in it.
- Display out the contents of hello.txt.
- Copy hello.txt into hello1.txt, hello2.txt, all the way up to hello100.txt. Obviously don't write 100 commands here.
- Now list all the files in abhiyaan.

**Bonus for part A: Nicer commands**

If you noticed, the above commands are kinda boring. Come on, when would you ever actually want to copy hello.txt 100 times? The true use of the bash and the command line come with shell scripting.

Here is a nice playlist of background music from a really cool video game. You should play it once when you get the time!But I want to download these tracks so that I can listen to them when iitmwifi cups. Thankfully, the People of the Internet have created a tool called yt-dlp that lets you do just this, from the command line!

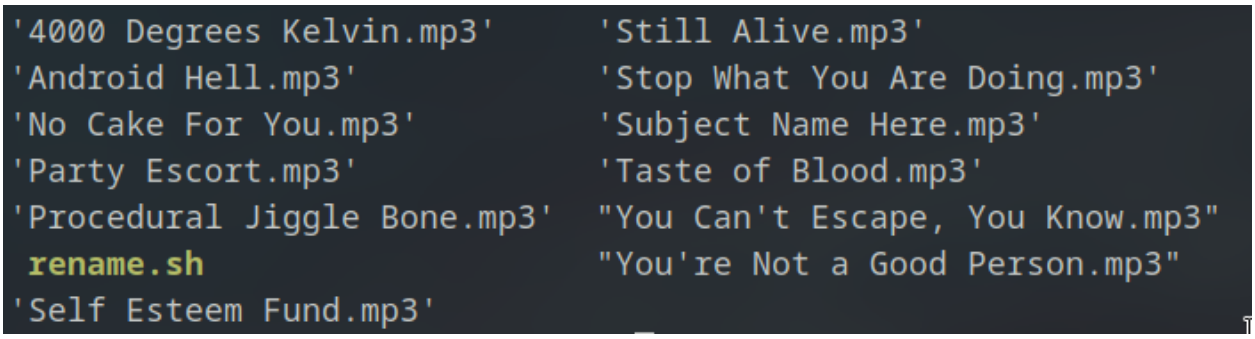
First, install yt-dlp and download the playlist as .mp3 files. By default, the tool downloads YouTube videos as, well, videos. So look at the options available to figure out how to do this. If all went well, you should get something like this:




File listing of downloaded tracks

But if you noticed, that's kinda ugly. Why is there so much extra text there? I just want the file names to be the names of the tracks.

**Write a shell script that renames all those files,so that you end up with something like this:**




File listing after renaming

This stack exchange might be a good starting point.

**Part B – Dangerous commands**

It's really easy to mess things up when you're using the terminal. Describe what happens if I run this command:

```
mkdir "~"
```

**How is this dangerous?**

Also, for fun, after you're done with this application, take a snapshot of your VM and run the "dangerous" version of the command on it.

## 1.4 udev rules

If you read about UNIX or Linux, you'd hear about the "everything is a file" philosophy. This means that everything (even physical devices connected to your system) is represented as a file. You can read and write to them like literally any other file on your system. A printer could be, for example, a special file which takes everything you write into it and prints it out on paper. A mouse is just a file which contains mouse movement data that gets updated as you move your physical mouse.

Try it now! Run `sudo hexdump /dev/input/mouseX` (where `X` may be `0` or `1` depending on your setup) and see the raw data that your mouse is sending.


This “always has been” a stale meme, but I just couldn’t help it.

All these files are present in a special folder in your root directory called `/dev` (for “device”). If you have a webcam, for instance, it will most likely be represented as `/dev/video0`.Allow your virtual machine to access a USB device connected to your laptop. Connect your phone to your laptop, switch on your VM, connect it to your VM through the USB settings menu on the VM.

Next, figure out what file name your laptop refers to your phone with. Once you find it, you’ll notice that it’s not very memorable – doesn’t really remind you of a phone.

What if we could tell linux to name it something more memorable, like `/dev/phone`? If we want to access stuff from our phone, it’s probably gonna be way easier to remember `/dev/phone` rather than what it is called by default.You can use udev rules to automate this task.

**Find the vendor and product IDs of your phone, and write a udev rule to automatically make `/dev/phone` refer to your phone each time your phone is connected.**

Here are some resources that you can use:

1. https://wiki.archlinux.org/title/Udev
2. https://www.reactivated.net/writing_udev_rules.html
3. https://wiki.gentoo.org/wiki/Udev

**This is a pretty difficult task, so even if you are able to get a partial understanding on how to proceed, very well done!**

## 2. ROS

Self-driving cars, as you can imagine, are complex and require a lot of software running in parallel. We need a way to allow different software to talk to each other easily, so we use another software "glue" that holds them all together – ROS.

**Part A:**

1. **Install ROS Noetic on top of your Ubuntu installation. The ROS wiki will be your best friend during this process.**
2. **Follow the ROS tutorials and create a catkin package in ~/abhiyaan called ros_abhiyaan.**
3. **Create a package called helloworld.**

Now let's write some actual code. ROS works with both Python and C++, but Python is easier to work with, and we currently use Python for ROS in our drive-by-wire system. The rospy tutorials will be very useful.

4. **Create a publisher node called sender that sends out either a -1 or a 1 (chosen randomly) onto a topic called readings (use Int32 as your datatype), at a time interval of 1 second. Display the results using rostopic.**

Let's now use the data that this node is generating.

5. **Create a node called integrator that subscribes to the readings topic and takes the sum of all the values it receives. It should publish this value as an Int32 to a topic called distance.**
6. **Create a node called maximum which monitors distance and keeps track of the highest value published in it. Each time a new maximum is detected, it should publish a string saying A new record! <value> where <value> is the new maximum, to a topic called message.**

Take screenshots of the outputs published by the above nodes.

**Part B – More ROS:**

ROS lets you publish all kinds of data onto topics – real numbers can use `Float` types, signed integers can use `Int` types, and so on.But what if we want to send two `Int`s at once? What if I'm sending some data which consists of both a `Float` and a `String`? We can use custom ROS message types for this purpose.

1. **Create a ROS Message type called `CheckedData` which consists of a `uint8` array called `rx` and a `uint16` called `checksum`.**

A common practice in communications systems is to use "hashes" or "checksums" to verify that data was not corrupted while being transmitted. One such technique is called the Cyclic Redundancy Check (CRC). You may remember seeing CRC in other sections in this application (or you might see it in the future when you reach it!)

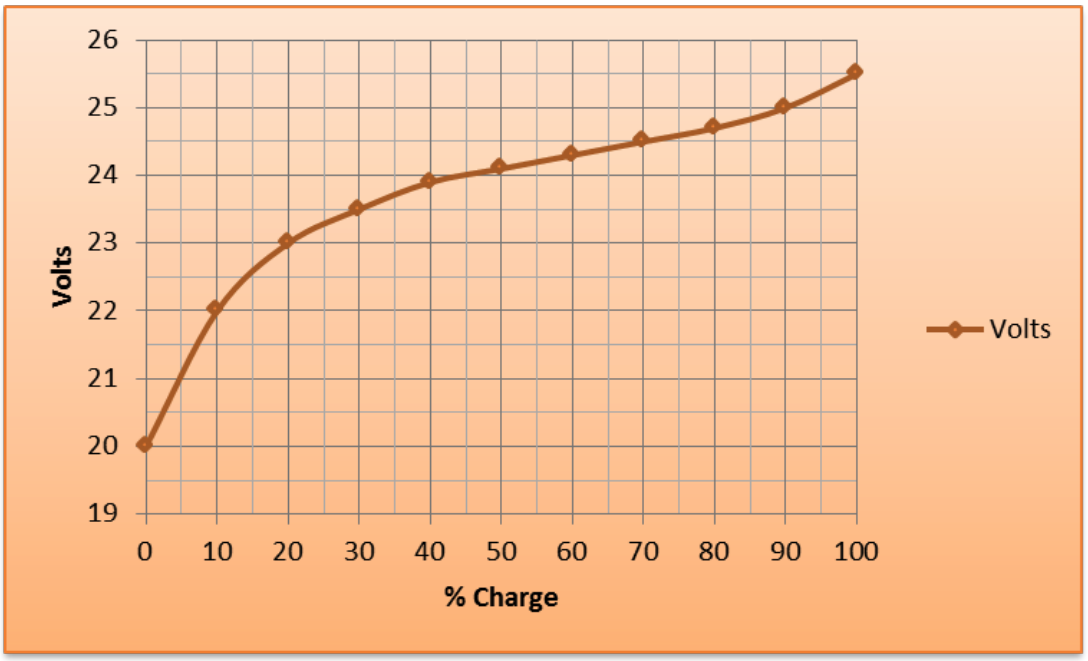
You can use the `crc` Python package (the one we asked you to install in the Linux section!) to create crc16 checksums.

2. **Create a node called `goodpub` which creates `CheckedData` objects. The `rx` parameter should contain 5 random bytes, and the `checksum` should contain the `crc16` hash of the 5 bytes. Make this publish at 1 Hz.**
3. **Create a node called `badpub` which publishes `CheckedData` objects to `rx_msgs` with 5 random bytes as above, but the `checksum` is always with `0`. Make this publish at 0.5 Hz.**
4. **Create a node called `verifier` which subscribes to `rx_msgs`, and verifies whether the `checksum` matches the `data`. It should publish either `ok` or `corrupted` to a topic called `crc_result`.**

# QUESTION-5

## A Battery Problem:

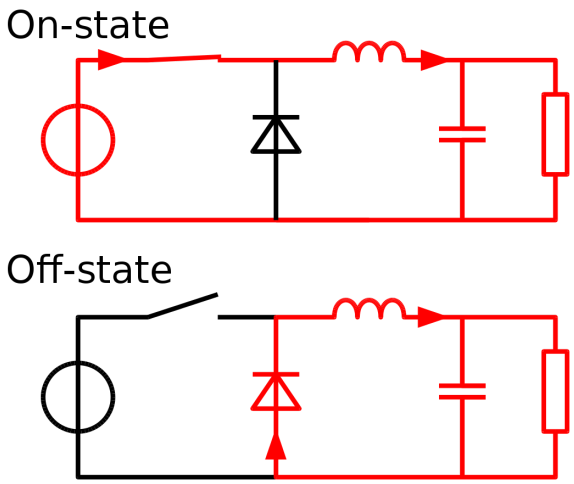
The battery's capacity is the maximum amount of charge that can be stored and discharged. % Charge/ State of Charge indicates how much the battery has been used and provides information about how much more energy it can supply.
Imagine a situation where you have a battery that has a voltage-capacity characteristic as shown in the figure:




Assume that the above battery has a capacity of 100AHr and can discharge completely.

a) **You have an actuator that requires uninterrupted 24V DC input but can tolerate supply between 23V and 27V. To ensure the device's safety, assume you will operate only in the region mentioned earlier. If the battery is fully charged initially and the actuator consumes 50W typically (assume the battery voltage is approximately the same in each 10% of the % Charge. Ex: For 90 -100 % of charge, battery voltage is 25.25V). Calculate how long the battery can supply power to the actuator.**

**b) Due to new plans, you must power a sensor that operates at 12V. While using a resistive divider would seem like a simple way to power the sensor, we can use a power conversion device whose operation doesn't depend on the load connected to it. Assume the power conversion device operates in the entire voltage range of the battery but has an efficiency of only 90% while providing a constant voltage of 12V at all times. How long can the sensor be used if it uses the same power of 50W?**




**c) The power conversion device mentioned above is called a DC-DC converter, specifically, a buck converter, since the device always steps down the voltage. The device works by switching the input supply on and off in a periodic manner.**
**Assume the switching frequency is so high that the capacitor voltage remains constant (Introspect on why this approximation is very valid) Understand and explain the basic principle behind the device.**

Have you ever glanced at the ratings on your phone charger's adapter? If you had, you would have noticed that the adapter charges your phone at a fixed voltage of 5V and supplies current in the range of 3-5 A, or even more when you want quicker charging.

Yes, let's design a circuit similar to a phone charger from our battery!

**PDSE1-S12-S5-M-TR** is a commonly used power regulator which takes in 12V and provides a stable power supply at 5V.
**PDSE1-S12-S5-M-TR** is an IC manufactured by CUI devices.

d) **Design a PCB in EAGLE which takes in 12V supply and provides 5V output using the PDSE1-S12-S5-M-TR converter. You can design the PCB based on the application circuit of the IC. Follow the EMC recommended circuit detailed by the manufacturer for bonus points. Use Phoenix connectors in the PCB to provide terminals for input and output. Design the layout once the essential connections are finalized. Check out for all the necessary design rules and follow them properly while designing your layout. (Use SMD resistors and capacitors.)**


e) **Now, find out why the suggested device PDSE1-S12-S5-M-TR is not suitable for phone chargers**.*(Hint: check out datasheet of the IC)*

A phone charger has much more complicated circuitry to ensure safe charging. Many things can damage the charger circuit and any other device that is connected to it. So there are various passive and active devices to protect the circuit.

f) **Formulate a circuit to ensure the safe operation of the voltage convert and implement it in the PCB. Look up for reverse protection diodes, ESD diodes, clamping diodes. Explain what made you choose the component that you chose.**

## References:

- Tutorials for EAGLE. You get free access to EAGLE with smail.
- ECAD model for PDSE1-S12-S5-M-TR and phoenix connectors: **ECAD-MODEL**
- Buck converters
- BMS

# Kudos! for PULLing through the application.

## To submit your answers:

https://forms.gle/9Q5ATAhjk5uSUfs76