General Questionnaire

1. India as a country is famously plagued with headache-inducing traffic. As someone who has lived in Banglore for 18 years, I couldn't agree more. The metro system in Banglore has over the years managed to control this issue to a large extent. I have come to see the hyperloop system as a for the lack of a better word modern iteration of the concept of metros.
   India is rapidly developing country and transport infrastructure needs to grow to meet the ever growing demand. An economically implemented hyperloop is in a good place to provide a novel solution to this growing demand. The lost productivity due to traffic jams is in the order of billions per annum, lessening commute time would also increase general productivity. An economically designed hyperloop model could after the initial capital and maintenance still effectively create a significant positive outcome for the economy. Not to mention the effect it would have in lowering vehicular emissions. If a complete hyperloop network were to be setup across the country and if it can reduce pollution on the order of EVs now on a larger scale it would be tremendous.

   In conclusion, I think the hyperloop would be a novel solution to modern transportation issues and it is feasible since I believe it likely bring more focus on innovation in transportation. The success of the first line would create demand for more hyperloop lines and this process compounds and will create a new market and a subsequent "race" for hyperloop innovation very possibly on a global scale.

2. Hyperloop to me seems like the most innovative CFI teams I know of. The entire concept is new and exciting, the pay-off would be immeasurable and being a part of a project like this is probably a dream come true for any passionate engineering student myself included. Career wise it obviously looks good on a resume and I want hands on experience doing work in my branch. I would also use it to learn more about team dynamics and management which will definitely be extremely useful in my career.

3. I want to work on the PCB and embedded software aspects of the Electrical Subsystem. I am passionate about low-level programming and working close to hardware.

4. EPFLoop has done research into optimizing their battery's cell count Paper
   MIT has an advanced electrical subsystem but idk if its an innovation Report

5. I am a DC in Math Club working on a 3D rendering mini-project and a DC is Programming Club under the CPP dev vertical. I can dedicate 18-20 hours a week comfortably and maybe 2-3 more hours depending on the week.

6. I am familiar with Python and C. I aim to learn CPP, Verilog and Matlab this semester. I have been programming in Python for 4 years from personal projects and minor scipting and C for over a year mainly from learning kernel development and to some extent course work.

   Personal OSdev Project

   Currently on dev hiatus doing research 😅

Question 1

Resisto can adjust the motor speed using the switch to create a pulse width modulated (PWM) signal. This setup needs Resisto to quickly flip the switch to create the signal.

This works because the PWM signal can control power delivered using the duty cycle. Though the flipping needs to take place quick enough for the motor to work as though on the average power.
This however is better than other methods like using a rheostat which would cause significant power depletion across the resistor.

This has several issues namely

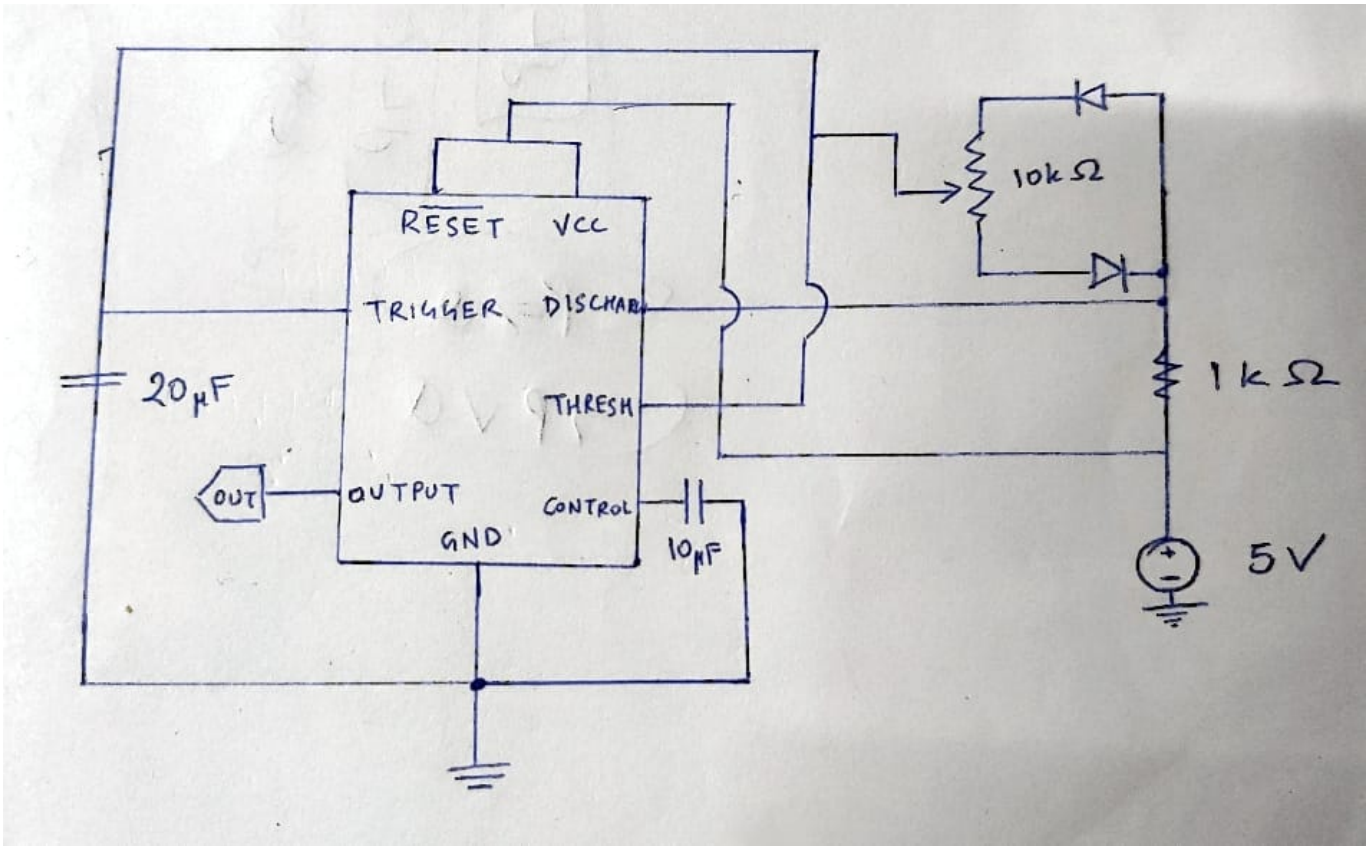
- Motor speed control is not precise
- Motor will not run smoothly
- Flipping switch is annoying

In both attempts we use a basic PWM signal generator using a 555 and a potentiometer to control duty cycle

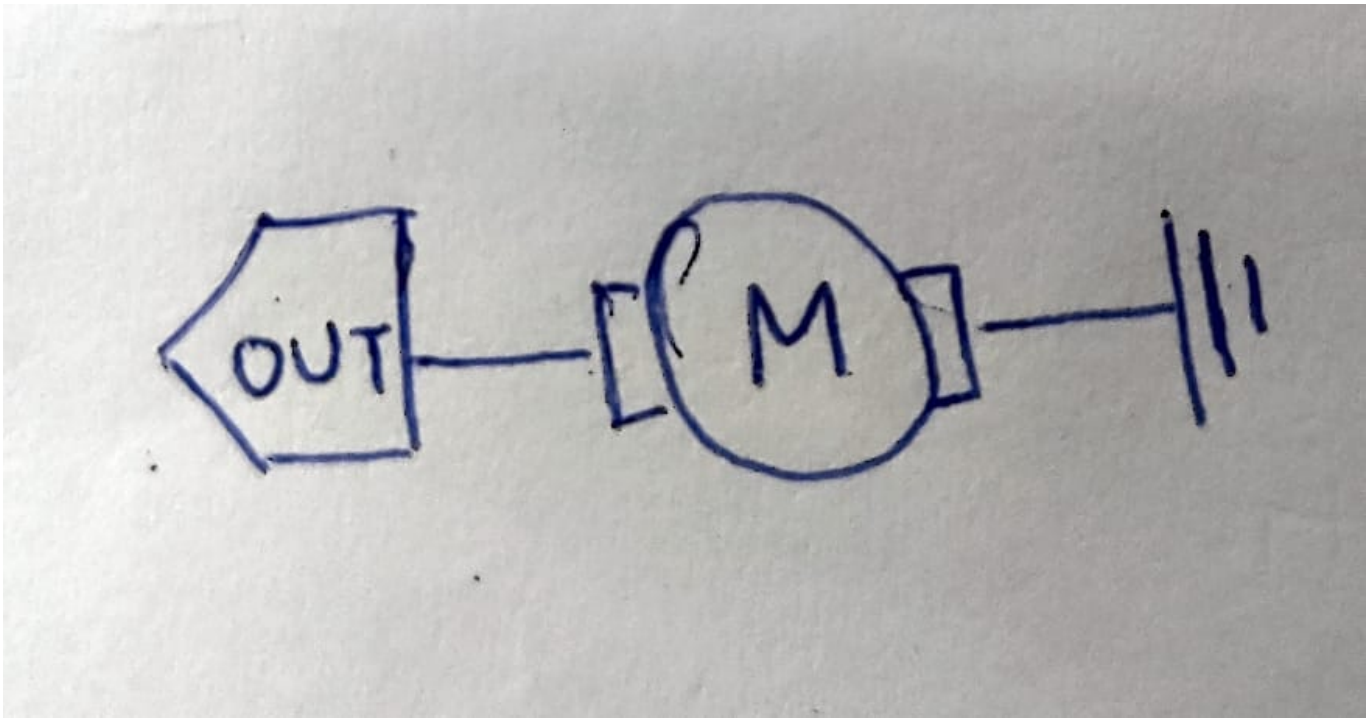
From the 555 data sheet output duty cycle (D) is

$$D = \frac{R_B}{R_A + 2R_B}$$

Where R_B is resistance of potentiometer



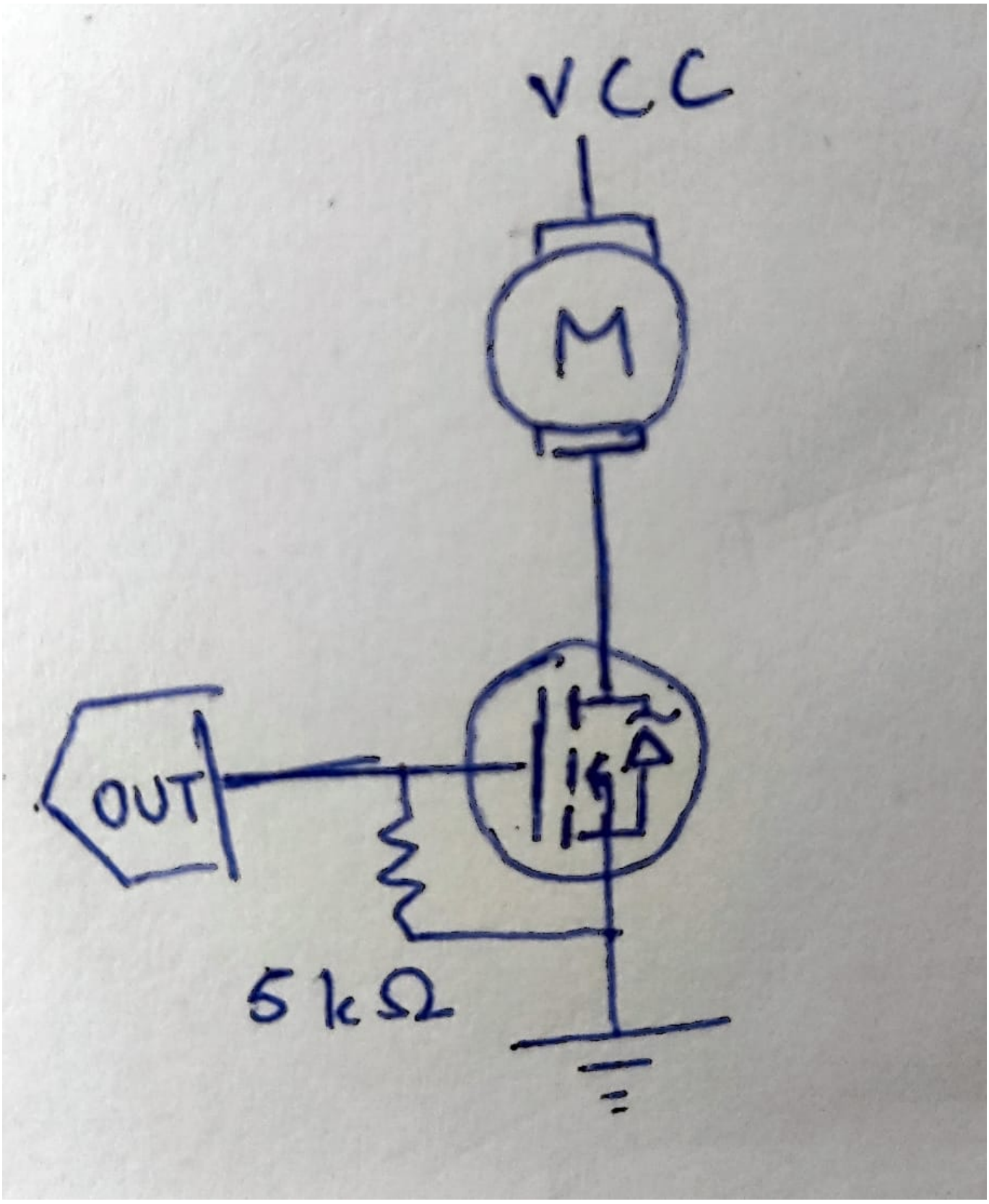
Dioden's first attempt might've been to directly use the output of the signal generator to run the motor



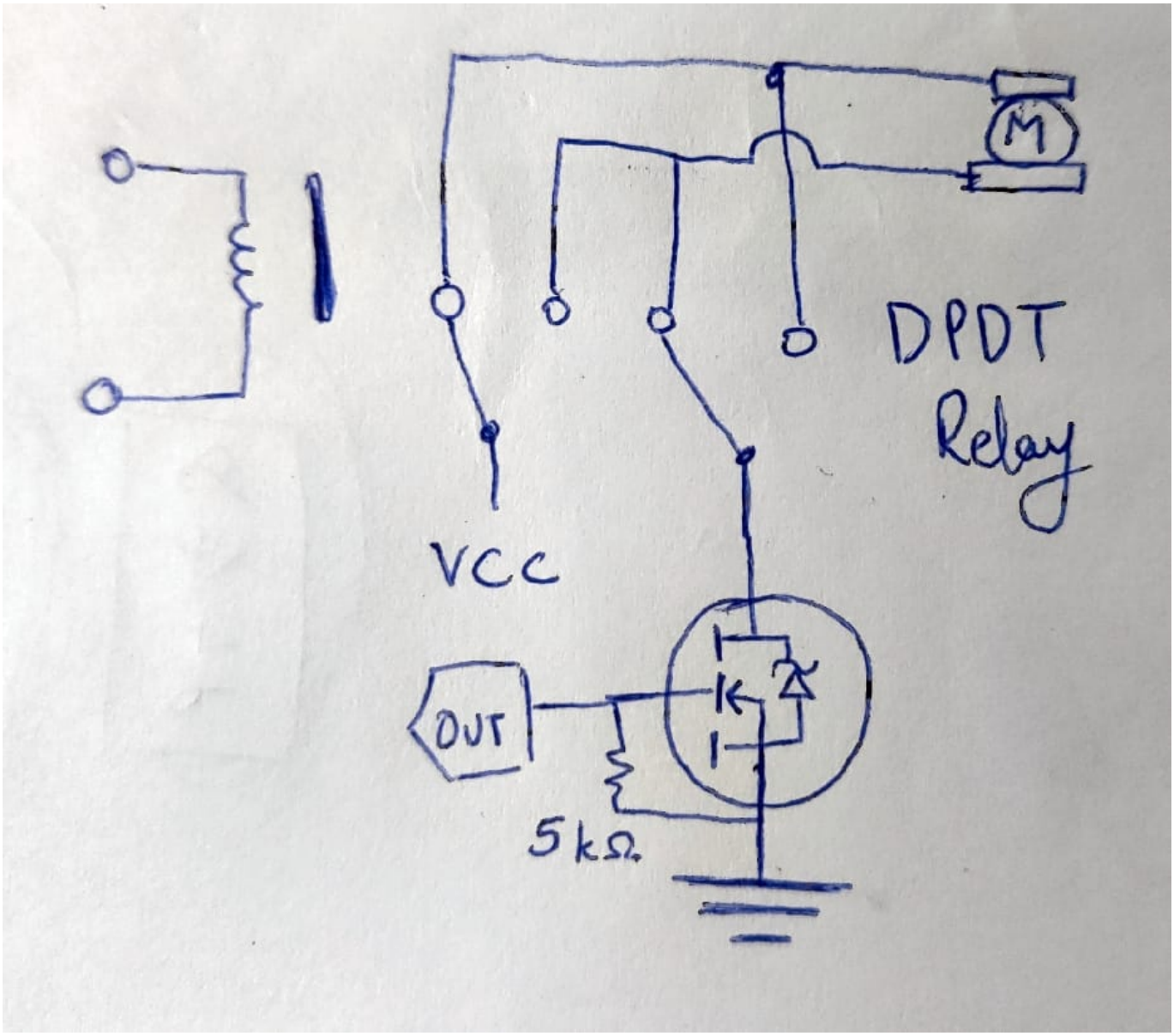
This isn't a good method since the signal may not be sufficiently strong, motor speed changing won't be smooth and we can't change direction.

In order to make it smooth we can connect the signal into the gate terminal of an NMOSFET which takes care of the first two problems by have source and sink connected

to an external voltage source



In order to change direction of the conveyor we need a DPDT relay to change the polarity of the motor. Activating the relay will switch the connections on the motor making it spin in reverse.



We should probably also heat sink the MOSFET to make sure it doesn't overheat.

The total cost comes out to

1	555 Timer	50
1	Potentiometer	15
2	Capacitors	40
2	Diodes	20
3	Resistors	60
1	MOSFET	60
1	DPDT	20
1	Heat Sink	150
		410

Sources:

- TI 555 Timer datasheet
- PWM Mosfet Circuit
- Mosfet heating

Question 2

1. The pre-charge relay is added to the circuit to make sure there is no in-rush of current on the load(inverter).
   If the pre-charge relay was not added to operate the circuit both main relays are closed and the capacitor charges up with no resistance in a very small time period. This sudden voltage spike across the inverter and accompanying inrush current can cause damage to the components.
   To fix this we add a pre-charge relay. We start by closing main relay(-) and the pre-charge relay, this is then simply an RC circuit that gradually charges. After a sufficient amount of time we can then open the pre-charge relay and close main relay(+) to begin usual operation.

2. Assume we pre-charge capacitor to 95% of max-charge (pre charge for 3τ), pre charge time T and max-voltage V_{fuse}.

$$T = 3RC$$

$$V_{\text{fuse}} = \frac{V}{R}$$

$$R = \frac{V}{V_{\text{fuse}}},\ C = \frac{V_{\text{fuse}}}{3V}T$$

3. That capacitor functions as a “DC-Link capacitor”. It is commonly used in DC-AC converters where it helps to keep the input voltage stable. This includes protecting it from spikes and ripple currents.
   The capacitance for the DC-Link capacitor depends on the input ripple currents and duty cycle. In general the minimum required capacitance is

$$C_{\min} = \frac{1000}{V_{\text{P(max)}}} \times \frac{d(1-d)I_{\text{out}}}{f_{\text{SW}}}$$

Where I_{out} is the output current, d is duty cycle, f_{SW} is switching frequency and $V_{\text{P(max)}}$ is maximum peak to peak voltage of the ripple current.

Sources:

- TI Pre-Charge application brief
- DC Link Capacitor

Question 3

The NCR18650BD data sheet puts the discharge end voltage at 2.5 V and capacity of 3 Ah. The LFP 26650 data sheet puts the discharge end voltage at 3.3 V and capcity of 2.5 Ah.

A 12s8p configuration of NCR cells gives discharge end voltage and capacity of 30 V and capcity of 24 Ah.

For achieving these parameters we need a 10s10p which gives 33 V and 25 Ah as its parameters.

Question 4

A)

- Traces:
  PCB trace refers to the network of copper, wiring, insulation and fuses that make up a board. The trace width is an import parameter to keep in mind while designing since passing high current through thin traces can lead to overheating and damage.

- Vias:
  Vias are holes drilled in a PCB with conducting walls. They allow the transfer of signals and power from one layer to another in a multilayer PCB. Via “covering” is usually done to make sure solder paste doesn’t get into it and cause short circuits. The simplest way being to simply cover it with a layer of solder mask.

- Polygon pours:
  Polygon pour is a technique used to create large areas of copper on the PCB. It is typically used for ground and power plane which provides low impedance connections to both. Copper having good thermal conductivity large areas of copper can dissipate heat effectively.

- Silkscreen:
  The PCB silkscreen is the text and images that we see on a PCB. The ink used is a non-conductive epoxy and comes in a lot of colours thought the commonly seen one is white. It is usually seen on the component side but having it on the solder side is not uncommon.

- Pads:
  Pads are copper areas on a PCB where we mount external components to the PCB. They come in two types through-hole or surface-mount. Through-hole pads are for components that have pins that go through the PCB like resistors. They are made by drilling holes in the PCB. For surface-mount components are directly mounted on the surface and doesn’t go through the PCB.

- Solder mask:
  Solder mask is a thin layer of polymer applied to copper traces and vias for protection

against oxidation and solder bridging. Green solder mask is commonly used as the industry standard most manufactures have been accustomed to it.

- Differential pair:
  Differential pair refers to a pair of signal traces commonly used it high-speed circuits to cancel out noise and enhance signal strength. Both traces carry the same signal but with opposite phase and the receiver samples the signal by taking the difference between the two signals. This has the advantage of removing any common noise and amplifying the signal.

B) The left most circuit is a 100nF capacitor tied across 5V and GND this functions as a "bypass capacitor" its main function is to suppress noise introduced through the power supply lines.

The next circuit is the one with the op-amp and USBVCC. The LP2985-**33**DBVR chip is 3.3V voltage regulator. If USBVCC is powered (USB is connected) it provides the 5V and the 3.3 V from the regulator. If VIN (DC power supply) is powered it will cause the op-amp to send a high signal which will open the P-Channel MOSFET (usually closed) and disconnect USBVCC.

Then there is the circuit with the power supply block. This block is the one that supplies VIN so it is the DC power supply. It is connected to a diode to make sure current doesn't flow from the board to the supply. It is then connected to the NCP1117ST50T3G chip which is a voltage regulator that gives out 5V.

Lastly the LED circuit glows green when 5V is high.

C) A thermistor is a resistor made of material with resistivity that is strongly dependent on temperature. They change resistance in response to a change in temperature which alters the potential difference across it which we can measure to calculate temperature. If we know the R/T curve for the thermistor we can use it to find temperature.

Steinhart–Hart equation is commonly used as a third order approximation for practical devices of large ranges of temperature.

$$\frac{1}{T} = a + b\log R + c(\log R)^3$$

where a, b, c are parameters

This circuit is a negative-feedback op-amp connected to the MSP's ADC. If we dont use the op-amp and directly connect ADC to the voltage divider the input will have high impedance will the ADC uses low impedance input. This configuration of the ADC makes sure that the voltage at all 3 terminals are equal to the voltage at the voltage divider but now the output terminal has low impedance. The voltage divider is used to make the input voltage within the ADC's reference voltage range (0-3.3 V).

The voltage at the divider V_{in} is given by

$$V_{\text{in}} = \frac{5R_{\text{therm}}}{R_{\text{therm}} + R_{\text{sense}}}$$

Assuming we are only dealing with 30-60 C temperatures (1.2-4 $k\Omega$)

$$V_{\text{in,max}} \geq \frac{5R_{\text{therm}}}{R_{\text{therm}} + R_{\text{sense}}}$$

$$R_{\text{therm}} \left(\frac{5}{V_{\text{in,max}}} - 1 \right) \leq R_{\text{sense}}$$

$$R_{\text{sense,min}} = \frac{68}{33} k\Omega$$

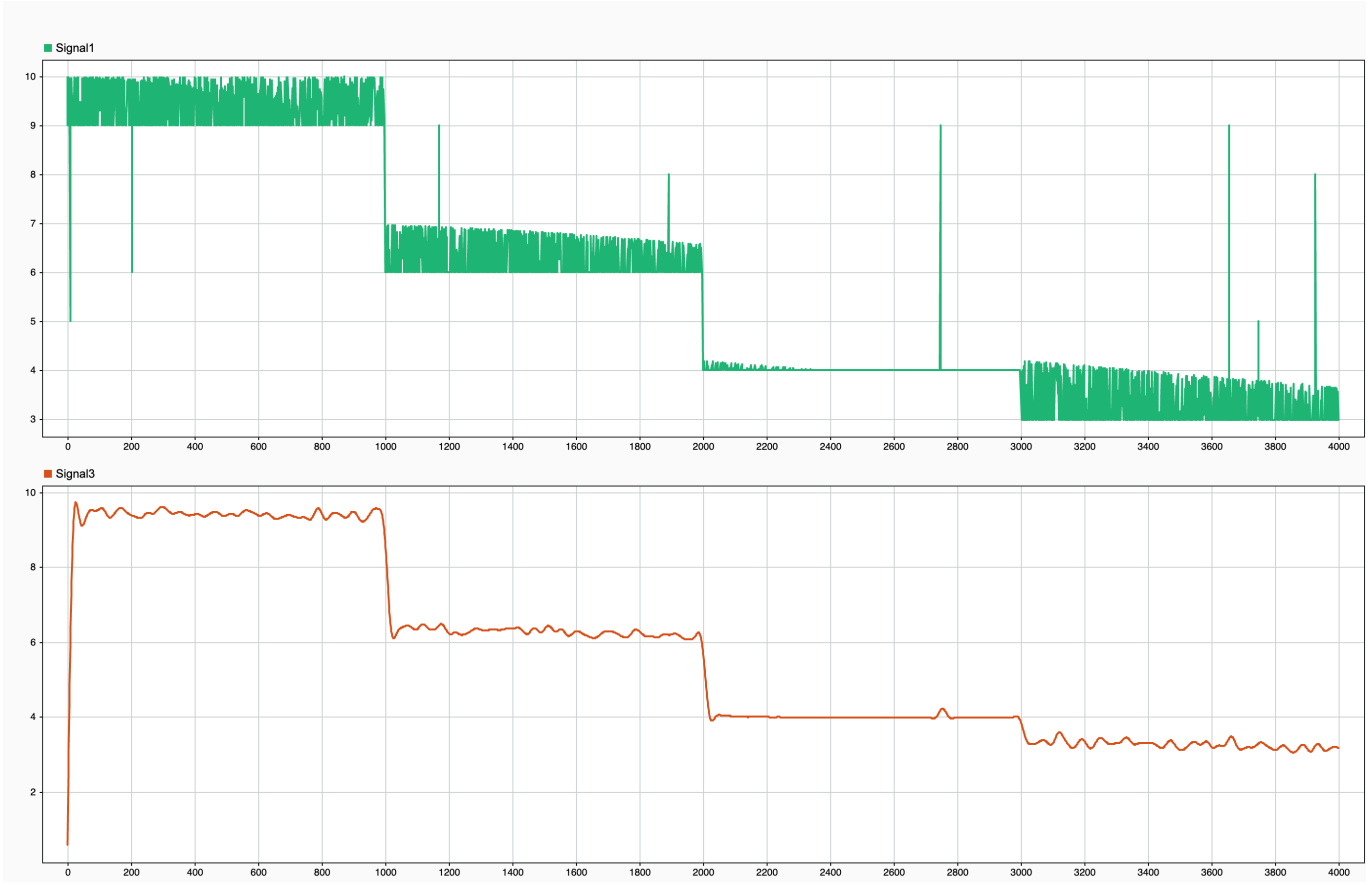
The ADC measures this voltage from which we can easily calculate resistance of the the thermister. The given thermistor has an exponential relation with temperature which we can use to find temperature.

$$R = 14.881 \exp(-0.0601T) k\Omega$$

This is a fit with $R^2 = 99.89$

Code for voltage to resistance: Github Link

Code for data processing: Github Link



Sources

- PCB GoGO
- Thermister Wikipedia
- Bypass capacitor Electronics SE
- LSP2985
- NCP117

Question 5

A relay is in essence an electrically operated switch. Mechanical or Electromechanical relay is one of the oldest types of relays. It is used when a circuit needs to be controlled by an independent low-power signal. This signal can be directed to an electromagnet which closes contacts. This low-power signal is the auxiliary signal.

Question 6

A) OSI has 7 layers namely

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

The function of the physical layer is to send and recieve bits over the network. This includes modulation, demodulation and channel coding.

The function of the data link layer is to ensure reliable communication between the ends of a link. Link layer offers frames to the physical layer for transmission. We achieve error control by attaching error-correction bits to the frame. At the receiving end the error-correction bits are used to check if the received message is accurate. The issues of multiplexing is handled in this layer. If the physical link is shared between several devices the sharing of the link is viewed as belonging to this layer. This is called medium access control (MAC).

The network layer ties together the links into a network. Each device is provided a unique network address at this link. Its primary function is routing packets between these addresses. Technically these 3 layers are enough for a packet switch. A distributed routing protocol is implemented in this layer to learn good good routes. These route computations are stored in a routing table at each switch. When the layer gets a packet the table is looked up and the packet is then sent to the link layer and queued at the appropriate output port.

The transport layer controls the reliability of a given link between source and destination through flow and error control. It provides acknowledgement of the successful data transmission and sends the next packet if no errors occurred.

The session layer provides services for dialog management between users. For instance

continuing the communication in-case of an error.

The presentation layer handles functions like data encryption and file conversions.

The application layer handles functions like file transfer and is the layer that directly interacts with the user.

CAN uses the physical, data link and application layers. The physical layer is the transceiver, the data link layer functions as the embedded controller and the application layer functions as a DSP.

B) Standard CAN bit fields have a 11 bit identifier (extended CAN has 29 bits).
If two nodes try to access the node at the same time access is implemented with a nondestructive bit-wise arbitration.
CAN messages have priority based on the identifier. The lower identifier has higher priority.
A standard CAN frame starts with a dominant SOF (Start Of Frame) followed by its identifier. Then there is a RTR (Remote Transmission Request) bit which differentiates between a data frame (dominant 0) or a remote request frame (dominant 1) then there is an IDE(IDentifier Extension) bit which denotes whether we are using standard or extended CAN frame format. Then there is a reserved bit that might be used in further updates of the frame standard. Then we have a 4 bit DLC (Data Length Code) segment which says how many bytes of data we are transmitting (0-8 bytes sending DLC 9-15 is physically possible but most controllers limit data to 8 bytes). We then have the data segment which ranges from 0 to 64 bits depending on DLC. Then we have 15 bits of CRC and then a recessive delim followed by a bit each of ACK, ACK delim(recessive) and a 7 bit EOF(End Of Frame all recessive) and 3 bits of IFS(Inter-Frame spacing all recessive).

C) Nodes access the CAN bus randomly. The bus's idle state is high which is the recessive bit.Nodes can monitor the bus while transmitting.
When nodes start transmitting they send a low SOF signal followed by their id and they keep monitoring the bus. If it reads a bit that doesn't match what it is transmitting then it stops transmitting and goes to receiver mode. This works since if it reads a dominant low when it is transmitting a high it means another node is transmitting with higher priority so to avoid a collision it stops transmitting. This is referred to as bus arbitration. After the end of transmission nodes that receive an accurate message transmit a dominant bit to overwrite the data frame's designated ACK bit (technically 2 bits but one is delim) indicating a successful transmission. If the bit is still recessive then the transmission failed

and after re-arbitrating the node tries to transmit again.

D) The CAN bus is a broadcast bus i.e all nodes receive all messages. Some nodes might care only about certain critical messages. For example if we have a battery controller connected to the bus we care about battery health messages more than the speed of the pod. If the controller receives the useless messages the software needs to process it. To fix this issue we can implement Hardware filtering in the CAN network.
Hardware filtering just filters out messages a particular node requires, this lessens the work software has to do. Not filtering messages uses up node resources for messages it won't process anyway.

```
FilterID : 0x15203424 | 0b10101001000000011010000100100
MaskID   : 0x1FAB677E | 0b11111101010110110011101111110
Accepted : 0x-------- | 0b101010_1_0_00_01__100_010010_
```

It accepts all identifiers of the form Accepted where each `_` can be 0 or 1. So there are a total of 2^8 or 256 accepted IDs.

i)

```
FilterID : 0x00000000 | 0b00000000000000000000000000000
MaskID   : 0x00000000 | 0b00000000000000000000000000000
Accepted : 0x-------- | 0b_____________________________
```

ii)

```
FilterID : 0x00000000 0b00000000000000000000000000000
MaskID   : 0x00000000 0b11111111111111111111111111111
Accepted : 0x00000000 0b00000000000000000000000000000
```

As far as I am aware it is not possible to make a CAN node accept no messages the minimum is 1. So I chose it to only accept the highest priority message.

E) The CAN frame has designated 16 bit (15+delim) bits of CRC (cyclic redundancy check) at the end of the frame before the ACK bit. CRC is a common error detection code used in networks and data storage. If any bit of the transmitted data is altered its CRC changes. So a node sends the frame with a pre-calculated CRC. If external stimuli like say noise alters the message the CRC will no longer match which lets the receiver know the message is invalid. Upon receiving an invalid message the receiver doesn't overwrite ACK which the

sender recognizes as saying the data integrity is compromised and tries to broadcast again after re-arbitration.

Sources:

- Anurag Kumar - Communication Networking: An Analytical Approach
- TI Introduction to CAN

Question 7

Data is in the application layer is just a bit stream with no inherent structure. In the transport layer data is broken into manageable segments with a header containing important information regarding the segment. This is called a packet.

Typically data flow would need to traverse more than 1 link to reach its destination node (else it would need to be fully connected) we use devices called switches to achieve this. It sits at the junction of links and moves the flow units between them. It de-multiplexes the input links and forwards them to the appropriate output link and then multiplex the output. For packet-switches (as opposed to circuit switch where multiplexing is done with physically partitioned bit pipe) this is a very fast function and must be performed on every packet.
Packet switching is important for the large scale hyperloop system since it is highly scalable, flexible, low cost and resource efficient.

From the DAQ to TelCU

```
Air gap data                        [Freuqent]
Brake actuation failure alert
Levitation failure alert
Battery overheating alert
Temperature data                    [Freuqent]
```

MCU to PCU & LCU

```
Stop the pod                        [Freuqent]
Brakes                              [Freuqent]
Start the pod                       [Freuqent]
```

Sources:

- Anurag Kumar - Communication Networking: An Analytical Approach

Question 8

A) A process is usually defined as a program in execution. It consists of its own section of memory, registers and logical program counter. In essence each process has its own virtual CPU.

A thread is sometimes referred to as a light-weight process. Each thread has a program counter for its current instruction, a stack and registers. Threads allow for multiple executions to take place in the process environment. Unlike processes threads in the same process space have access to each other's memory. This ability is crucial for certain applications.
Thread creation is also significantly faster (10-100x) than process creation since they are more light-weight.

Processes are scheduled by the system's process scheduler and each process has a unique process id (PID). The OS allocates CPU time to each process based on process priority and the OS's scheduling algorithm. Threads meanwhile are scheduled by the parent process's thread scheduler. Threads like processes also have a unique id (thread id TID).

When CPU needs to switch processes it needs to switch process context like loading saved state into the registers and this switching has higher overhead while threads being lighter have little to no switching overhead compared to a process.

Communication between processes needs sophisticated techniques for inter-process communication (IPC) and is slower than inter-thread communication (ITC) since they share common memory address space. This however introduces the risk of race conditions and data corruption in ITC if proper synchronizations techniques are not used.

Bonus: Intel Pentium 4 introduced multithreading (hyperthreading) to the x86 processor. This allowed CPUs to store the states of two (or more) threads and allowed the CPU to switch between them on a very small time scale.
This feature can increase the execution speed in general. For example when the current process requests uncached data from memory which takes several clock cycles to complete the CPU can just switch to another thread to use the unused computing resources.
The OS sees the number of "logical threads" which is the total number of threads that can be scheduled and executed by the CPU. Since task manager simply displays the thread

count seen by the OS the counts different from the one reported by the CPU manufacturer which is the physical thread count.

B) This is a naive thread interpretation

```
Start program
Create a global balance variable
Start 2 threads
Let each thread add or subtract some amount from the balance [Repeat 2000 time
End program
```

There wouldn't be any errors 'persay' but there will be a major bug in this implementation. (Further explained in BONUS)

Threads are prime jurisdiction for ✨ Murphy's law ✨, thread concurrency can lead to a lot of undefined behavior since they share memory and other resources so we implement techniques to handle such resources keeping threading in mind.
Some techniques used in software development to handle resources altered by different threads parallely are

- Semaphore
  A semaphore is simply a variable that can store a positive integer. A thread can interact with it using 2 atomic methods up and down. Up simply increments the semaphore, down tries to decrement the semaphore if it is already 0 the thread gives CPU to another thread otherwise it decrements.

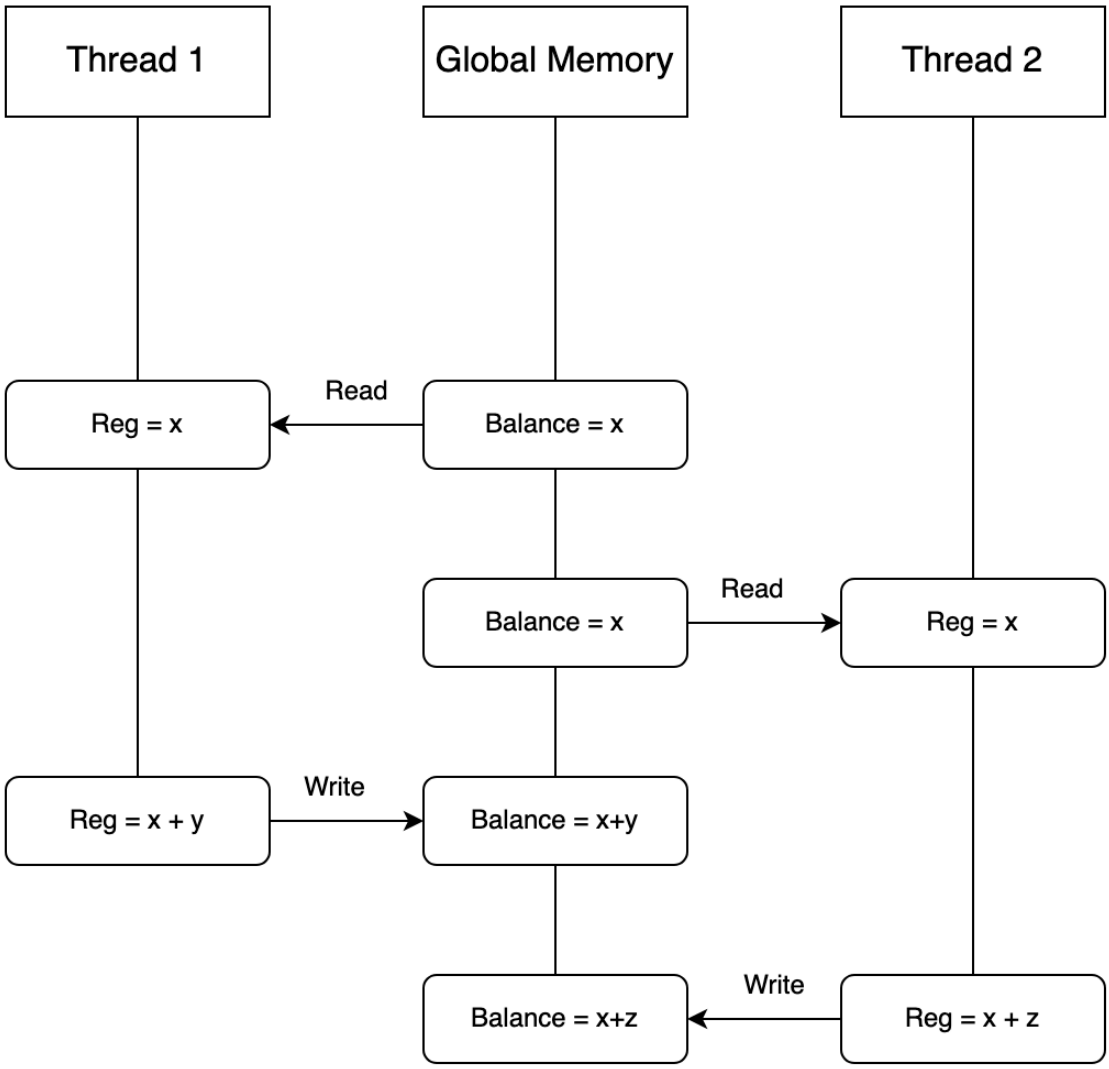
- Mutex
  In this technique we use a shared variable called the mutex that is either a 0 or 1. If a thread is currently using the resource it is protecting the mutex is 1 and 0 otherwise. It is implemented with 2 methods lock and unlock. The thread calls unlock before using the resource if the mutex is 0 it changes it to 0 and then hands control back to the thread which then calls unlock after using the resource which sets the mutex to 0. If the mutex is 1 then the method gives the CPU to another thread and checks again when it receives CPU control. The two mutex methods are atomic in order to not cause race conditions.
  Ex. Above question (below answer?)

Bonus
Naive Thread

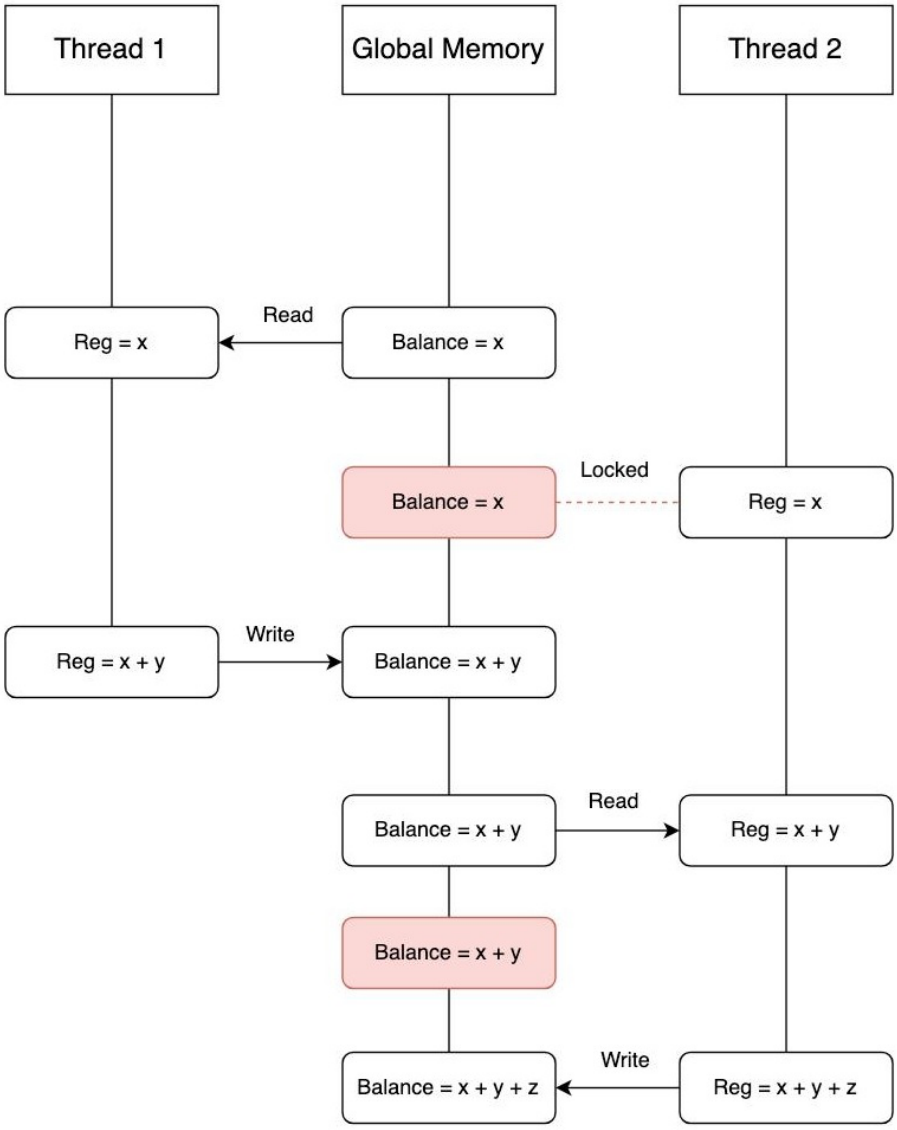
[Mutex Thread]

Discrepancies occur in the naive implementation. When we want to change the value of a global variable by say incrementing/decrementing it we first copy the value of the variable to one of the thread's registers. The value in the register is then changed and saved back into the global variable.

Discrepancies can happen here if one thread copies the global value and while it is processing it another thread copies the value too. So instead of having both threads' changes applies to it only one of them is.



We can fix this discrepancy by implementing a mutex lock on the balance. Which simply means only one thread can have access to it at a time.



Sources:

- Andrew S Tanenbaum - Modern Operating Systems
- IBM AIX docs
- Multithreading wikipedia