Project Altair Recruitment

Name: Rumman Adib

Student ID: 210041131

Department: CSE

Github Repository: https://github.com/Rumman023/Project-Altair-Recruitment

Theoretical Part

TASK 1:

For my self-driving delivery robot, as the mechanical and electrical engineering has been done already, I have to decide and implement a software architecture that can aptly support the gear motors and PWM-controlled motor drivers for wheels, servo motors for steering, a rotary encoder, GPS, IMU, and a Stereo Camera.

The architecture which I'm inclining to use, will consist of both the Microcontrollers and SBC, so it can be called Hybrid Control Architecture. This architecture will focus on precise control and efficiency of every component of the robot, that's why both Microcontrollers and SBC will be used. The microcontrollers will control all the motors, motor drivers and encoders, where the SBC will serve as the CPU for all the sensors creating a sensor fusion algorithm.

Microcontrollers: In a high precision demanding architecture, having an option to control the speed of gear motors is very much important. An efficient technique of such control can be the 'PWM' or Pulse Width Modulation. An Intel 8051 microcontroller can be used to produce the PWM wave with an integrated timer, where the varying width of this PWM wave can control the speed of motors. A PIC16F877A Microcontroller is to be used to interface the rotary encoder. When the rotary encoder disk moves in any direction with the rotating shaft of any motor or any object then the voltages are generated in the form of pulses and these pulses can be efficiently observed by the microcontroller. Thus the rotary encoder can be efficiently decoded. Therefore, this architecture uses 2 microcontrollers – 1 Intel

8051 Microcontroller and 1 PIC16F877A Microcontroller for all motor related works.

Single Board Computer (SBC): Raspberry Pi is known as the most cost-effective SBC with enough computational power and a faster and more efficient CPU and GPU. For the architecture, I would use a Raspberry Pi SBC which can simultaneously interface with GPS, a stereo camera and an IMU because of its multi-dimensional usage ends. With compatible modules and necessary libraries, Raspberry Pi can serve as the CPU of such sensors and then the later part of high-level decision-making procedures. Proper Sensor fusion algorithms can be implemented to our SBC to merge the data received from all the sensors, making it accessible for the robot to have adequate understanding of its state and surroundings.

Justification: As microcontrollers are typically more suitable for all real-time control of motors, our standalone architecture suits perfectly for this cause. It allows us to monitor and control all the changing condition in real-time which increases the preciseness of the implemented architecture. As our architecture incorporates the solely dedicated SBC such as Raspberry Pi to all the sensors, it increases the scope of integrating various advanced technologies in any conditions. Again, in any kind of hazardous situation where any component fails, having a separate microcontroller for motors and SBC for sensors can strengthen the whole architecture as it is less likely to affect other parts of the system.

References:

1. https://www.electronicshub.org/pwm-based-dc-motor-speed-control-using-microcontroller/
2. https://www.mas.bg.ac.rs/_media/istrazivanje/fme/vol37/1/02_nvukovic.pdf
3. https://microcontrollerslab.com/rotary-encoder-module-interfacing-pic/#:~:text=When%20the%20rotary%20encoder%20disk,these%20pulses%20with%20so%20efficiently.

4. https://en.wikipedia.org/wiki/Sensor_fusion

5. https://www.makeuseof.com/microcontrollers-single-board-computer-differences/#:~:text=The%20Differences%20Between%20Single%2DBoard,chip%20with%20far%20fewer%20resources.
6. https://www.arrow.com/en/research-and-events/articles/sbc-needs-specific-to-robotics

TASK 2:

When Arduino Mega and Arduino Nano are put in a single PCB, the most optimal and efficient communication protocol should be chosen to establish communication between these two microcontrollers. Among the most used communication protocols- which are: UART, I2C, SPI; most suitable protocol should be chosen considering different factors like Data Transfer Speed, Complexity, Implementation and Hardware specifications. Considering all of these factors, I would choose the UART communication protocol for its simplistic and minimal approach of configuration.

Why UART?

The answer for this question simply lies in the advantages of this protocol over the rest of the other protocol. UART or Universal Asynchronous Receiver Transmitter is a serial communication protocol meaning data bits are transferred sequentially, one after the another. For interfacing the UART communication, two wires are needed, one is the Tx (D1) pin and the second one is the Rx(D0) pin of the Arduino board. Tx pin transmits data to devices and Rx pin receives that data. The advantages of this protocol include: it is simple to operate and implement, it doesn't need any clock signal, it prevents data loss once the Baud rate is set within the 10% of limit, it is half duplex meaning the devices cannot transmit and receive data at the same time. Also this protocol doesn't require any extra component but it has less data transfer rate than the I2C and the SPI protocol.

UART vs I2C vs SPI:

Although the I2C and SPI protocols allow multiple devices access in their protocols, it increases complexity of both operation and implementation. UART doesn't require any additional components but the SPI requires dedicated hardware pins like MISO, MOSI, etc. I2C and SPI are generally faster than UART, but some of the disadvantages of these protocols include their increasing circuit complexity with additional master/slave setups, and is only able to operate in half-duplex or full-duplex. UART can also be used in long distance communication compared to the other two protocols, and it is a simple and reliable protocol than the rest of the others because its simple implementation with no clock signals.

Code Snippets:

References:

1. https://www.instructables.com/Arduino-I2C-and-Multiple-Slaves/
2. https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/
3. https://www.seeedstudio.com/blog/2019/11/07/arduino-communication-peripherals-uart-i2c-and-spi/
4. https://linuxhint.com/arduino-communication-protocols/

TASK 3:

A)

DC Motor: DC Motor is a typical electric motor (brush or brushless) that converts the electrical energy (DC Current) into mechanical energy to be used in shaft rotation.

Stepper Motor: Stepper Motor is a typical brushless electric motor where the rotor of the motor is divided into equal and discrete steps for each full rotation, thus obtaining the name “Stepper Motor”.

Usage: DC motors are commonly used in toys, home appliances, computers, lifts, automotive architectures, EV, etc. On the other hand, Stepper motors are mainly used in robotics, printers, hard disc drives etc. It is to be noted that, DC motors are designed to operate continuous tasks where Stepper motors are designed to operate in short term discontinuous tasks.

Handling of Motors: An easy way to control the speed and rotation of DC motors is to regulate the supply voltage with pulse width modulation (PWM) because the speed of a DC motor is directly proportional to the supply voltage. There are many ways to generate the PWM signal for the motor, and 555timer is one of them. Also, a H-Bridge Motor Driver is used to reverse the motor’s direction by changing the applied voltage poles to the motor terminals. On the other hand, there are some major modes of Stepper Motor controls- Wave Drive, Full Step, Half Step, Microstep. With specialized driver circuit, stepper motors can generate required sequence of pulse for each step of rotation.

B)

There are different types of motor driver chips or ICs used in different types of motors according to their functionality and purposes. The most commonly used motor driver ICs include: L298N, PCA9685, A4988, IR2104, etc.

DC Motor Driver: The L298N module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A. The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors simultaneously.

AC Motor Driver: The IR2104 IC is a half-bridge driver which accepts low-power input to output high-current drives. It feeds the gate of a high-power transistor like a power MOSFET. In addition, the IR2104 gate driver functions as a level shifter and a power amplifier for AC induction motors.

Stepper Motor Driver: The A4988, with built-in translator, is a complete microstepping motor driver. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A. The A4988 includes a fixed off-time current regulator which has the ability to operate in slow or mixed decay modes.

Servo Motor Driver: Where large number of servo motors are used such as robot arm, hexapod and robots, The PCA9685 servo driver module is used. It increases the number of PWM output of its connected microcontroller. Using only two pins, a 16 free-running PWM outputs can be controlled.

C)

Definition: Pulse-width modulation (PWM) is a technique of controlling the average power delivered by an electrical signal to a load.

How it works?

The idea is fairly simple. This technique uses a rectangular pulse wave whose pulse width is modulated resulting in the variation of the average value of the waveform. The term "duty cycle" of 50% and resembles a "square" wave. This duty cycle is expressed in percentage, where the ratio describes the proportion of "On" state compared to the total time of one full cycle. The average power delivered to a load can be controlled by regularly changing the duty cycle of the PWM signal.

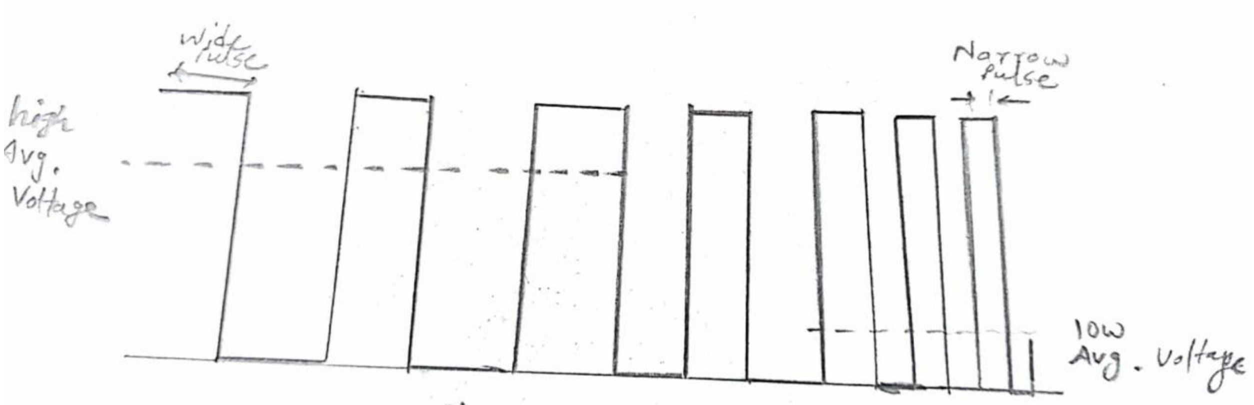


Fig: PWM

How it is used to control motor speed?

An easy way to control the speed of a DC motor is to regulate the supply voltage with pulse width modulation (PWM) because the speed of a DC motor is directly proportional to the supply voltage. The basic idea behind this is that it switches the supply voltage ON and OFF very rapidly. By adjusting the length of the ON/OFF pulses, the voltage can be set to anywhere between 0V and the maximum voltage.

D)

Definition: An encoder is feedback provider device that converts motion into an electrical signal that can be read by some type of control device in a motion control system.

What is encoder feedback?

It is a method of providing real-time information of a motor's speed, rotation, direction by converting the motion into electrical signal. This helps to gain precise control over a motor. To have accurate motor feedback, the encoder should be selected and implemented carefully.

Implementation:

To implement a motor feedback in a rover, I would choose the right kind of encoder that goes along with my rover's functionality. After choosing the suitable encoder, it has to be integrated carefully into the software architecture. I would carefully go through these important steps like wiring and programming the encoder with the microcontroller to get important encoder data. From the collected data, I would try to gain precise control over the motor by some trial and error procedure.

So this is the method I would choose, if I were to implement the motor feedback with encoders for precise rover movements.

References:

1. https://www.tutorialspoint.com/difference-between-stepper-motor-and-dc-motor#:~:text=Stepper%20motors%20have%20incremental%20motion,DC%20motors%20have%20continuous%20motion.&text=Stepper%20motors%20give%20slow%20response,response%20than%20a%20stepper%20motor
2. https://www.thomasnet.com/articles/machinery-tools-supplies/stepper-motors-vs-dc-motors/
3. https://www.circuitbasics.com/introduction-to-dc-motors-2/
4. https://www.polycase.com/techtalk/electronics-tips/how-to-control-stepper-motors.html#:~:text=Fundamentally%2C%20the%20basic%20method%20of,the%20needs%20of%20their%20applications
5. https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/#:~:text=L298N%20Driver,and%20explain%20how%20it%20works.

6. https://www.allegromicro.com/en/products/motor-drivers/brush-dc-motor-drivers/a4988#:~:text=The%20A4988%20is%20a%20complete,V%20and%20%C2%B12%20A.
7. https://store.fut-electronics.com/products/pca9685-16-channel-servo-pwm-driver-i2c-interface#:~:text=The%20PCA9685%20servo%20driver%20module,16%20free%2Drunning%20PWM%20outputs.
8. https://www.wellpcb.com/IR2104.html#:~:text=The%20IR2104%20IC%20is%20a,shifter%20and%20a%20power%20amplifier.
9. 1.https://www.digikey.com/en/blog/pulse-width-modulation#:~:text=Pulse%20width%20modulation%20turns%20a,to%20when%20it%20turns%20off.
10. https://en.wikipedia.org/wiki/Pulse-width_modulation
11. https://www.circuitbasics.com/introduction-to-dc-motors-2/
12. https://www.encoder.com/article-what-is-an-encoder#:~:text=Simply%20put%2C%20an%20encoder%20is,count%2C%20speed%2C%20or%20direction.
13. https://community.robotshop.com/forum/t/rover-5-encoder-feedback-loop-speed-to-encoder-pulse-duration-conversion/4862

TASK 4:

Different Types of Microcontrollers: Arduino Mega is an open source 8-bit development board based on Atmega2560 AVR microcontroller. It is popularly used in various home projects, robotics, Iot projects, 3D printers, and various other real world applications. The ESP32 is a System On a Chip (SoC) general purpose microcontroller with an extensive peripherals including WiFi and Bluetooth wireless capabilities. It can perform as a master or slave device to reduce communication stack overhead as the main application processor. STM 32 is a 32-bit microcontrollers based on the Arm Cortex processor which is designed to offer versatile periphery to MCU users. It offers a large number of serial and parallel communication peripherals which can be interfaced with all kinds of electronic components.

1.Motor Control: If a robot has multiple motors in its embedded system, the most suitable microcontroller can be the STM32's STM32F103C8T6- bluepill because it is the most budget friendly and compatible option. The Blue Pill is quite cheap, which makes it a good choice for all the aspirant students or hobbyist. As it is based on 32-bit Arm Cortex processor, it has a powerful backup compatibility for the Arduino ecosystem which helps greatly for working with it. It has powerful processing capabilities featuring an extensive peripheral such as multiple PWM channels. These features make it easy to incorporate multiple motors in a robot's system using STM32 blue pill. Also it offers Real time capabilities which is essential to control multiple motor control application in a robot. These are the few reasons why the STM32 Blue Pill is superior than others in handling multiple motors in an embedded system of a robot.

2.Communication: As ESP32 microcontroller has built-in WiFi and Bluetooth features, it makes superior than the other microcontrollers in terms of long range communication. The ESP32 has dedicated library for long range communication, one of which is the LoRa, which enables long-range data transmission with low power consumption. This technology can achieve data rates between 0.3 kbit/s and 27 kbit/s depending upon the spreading factor. Also ESP32 has good community support so if any problem is faced during building a project, one can easily ask for help in their dedicated community. These are the few reasons why the usage of ESP32 is justified in the matter of long range data communication.

3.Real-Time Operation: The STM32 microcontroller is best when it comes to real time operation feature. The microcontroller can precisely produce accurate clock pulses and a plethora of timers with flexible using scope. This microcontroller can support various Real time operation systems (RTOS) including FreeRTOS, which is free and open source RTOS. It offers task management, inter-task communication, synchronization mechanisms, memory management, etc. RTOS ensures that the tasks has to be executed concurrently, and the execution time of task has specified time frame. This approach can be easily integrated in a drone flight controller using the STM32 development environment.

4.Sensor Integration: The Arduino Mega 2560 can be a great choice for a home automation project and integrating different sensors relevant to the project. The Arduino Mega 2560 supports different sensor libraries such as : DHT sensor library, IRremote library, PIR sensor library, etc. The Passive Infrared Sensor (PIR Sensor) is an electronic sensor that measures radiating infrared (IR) light from objects from its view port. This technology is used in outdoor motion detector systems in many homes. It is easy and budget friendly to implement the PIR Sensor in an Arduino Mega 2560, thus it makes superior than all the other microcontrollers in the context of Sensor Integration.

5.Power Efficieny: If power efficiency in a battery-operated remote sensor node deployed in a forest is a concern, then the ESP32 is undoubtedly the best choice. The ESP32 has five low power modes, these are: modem-sleep, light-sleep, deep-sleep, hibernation, and power off. These modes allow the system undergo in a low power consuming state when the system isn't being used, thus allowing energy to be conserved. This feature is essential in any battery operated applications. As this system has inherent WiFi and Bluetooth system, it helps to conserve energy by monitoring the unnecessary nodes in a forest and cut off the power supply to those specific nodes. These are the few features that easily justifies ESP32 as the best microcontroller to be used in a battery-operated remote sensor node deployed in a forest.

References:

1. https://microdigisoft.com/stm32-blue-pill-vs-other-stm32-boards-comparison/#key-advantages-of-the-stm-32-blue-pill
2. https://dl.acm.org/doi/abs/10.1145/3436286.3436750
3. https://www.hackster.io/news/long-range-wifi-for-the-esp32-9429ab89f450

4. https://en.wikipedia.org/wiki/LoRa#:~:text=LoRa%20enables%20long%2Drange%20transmissions,depending%20upon%20the%20spreading%20factor.
5. https://www.ti.com/lit/wp/spry238/spry238.pdf?ts=1690457798494&ref_url=https%253A%252F%252Fwww.google.com%252F#:~:text=A%20growing%20number%20of%20MCU,that%20need%20to%20be%20supported.
6. https://www.instructables.com/PIR-Motion-Detector-With-Arduino-Operated-at-Lowes/
7. https://www.arrow.com/en/research-and-events/articles/esp32-power-consumption-can-be-reduced-with-sleep-modes#:~:text=The%20ESP32%20has%20five%20low,based%20on%20active%20processing%20capabilities.

TASK 5:

Sensors are a vital component in any robotic system as it provides vital sensory data including size, orientation, velocity, distance, temperature, weight, etc. In a Martian outdoor environment, A robot must have these sensors: Stereo camera, LIDAR, IMU, Tilt sensor, Thermal sensor, Gyroscope, encoders, IR distance sensors etc.

Explanation of These Sensors:

For understanding the environment: The most vital sensor for any robot is the **stereo camera** as it provides the visual data about its surroundings. As Stereo camera's can access more data, it produces less occlusions. Also it can emulate human binocular vision and thus can gain the ability to perceive depth. Having a camera mounted is to literally have all the necessary information around its surroundings which can help it to detect obstacles, avoid bumps, perform visual odometry, etc. Then next most important sensor to have is the **LIDAR** sensor, because it uses laser beam to detect object and create a detailed 3D map of the robot's environment. This technology assists robots with real time information to navigate their surroundings by providing object perception, object identification and collision avoidance. To understand the orientation of the robot in the martian environment, the **IMU** must have to be integrated. This sensor provide estimation of the robot's orientation in space, by integrating multi-axes, accelerometers, gyroscopes, and other sensors to provide estimation of an objects orientation in space. Therefore it helps in navigating, path control by estimating the robot's motion and orientation. **Ultrasonic Sensors** are used to supplement the LIDAR data for close-range object detection by measuring short range distances. **Gas sensors** are also used to classify any potentially harmful gas's existence in the air.

For controlling the movement: A **Wheel encoder** is a sensor that detects rotation angle or linear displacement and is used in high speed and high accuracy demanding systems. The technique of controlling the motor rotation by detecting the motor rotation speed and rotation angle using an encoder is called feedback control. Also the **Proximity Sensor** and **Ultrasonic Sensor** both help to give real time information to the LIDAR sensor for better understanding and control over the robot's movement control.

Resources:

1. https://www.phase1vision.com/blog/lidar-helps-nasa-explore-mars
2. https://www.wevolver.com/article/sensors-in-robotics-the-common-types
3. https://www.e-consystems.com/blog/camera/technology/what-is-a-stereo-vision-camera-2/#:~:text=A%20stereo%20camera%20is%20a,the%20ability%20to%20perceive%20depth.
4. https://www.akm.com/global/en/products/rotation-angle-sensor/tutorial/role-encoder/#:~:text=An%20encoder%20is%20a%20sensor,speed%20and%20with%20high%20accuracy.

TASK 6:

Logical Part

TASK 1(a):

Code:

https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_Rumman%20Adib_Week%201_Logical_Task%201_A.cpp

Explanation:

1.Problem Diagnosing: Counting the number of connected component is enough for this task. If the count of the connected component in a bi-directional graph exceeds 1, then the graph surely doesn't have a valid path to visit all of its nodes/vertices. It can be easily demonstrated that, if we can visit all the vertices from a single source node in a graph, then the graph must have only one connected component.

2.Solution Approach: In my code, I counted the number of connected components using the number of DFS calls the algorithm will give. Because, the code has a

Boolean visited array all of which are initialized with 0, meaning none of the nodes have been visited. The algorithm executes a DFS from the source node and it will mark itself and all the other visitable nodes as ‘can be visited’. This iteration will increment the connected component counter. The algorithm won’t call next DFS until a node/vertex is found, which can’t be visited from the source node. If such node/vertex is found, another DFS call will be executed on that newly found vertex and thus incrementing the number of connected component. This algorithm will be executed until all the nodes/vertices are marked as ‘can be visited’. If the number of connected component is found to be greater than 1, this means that the graph doesn’t have a valid path that exists to visit all the nodes/vertices. Thus the output will be ‘false’. Otherwise the output will be ‘true’, meaning in the given graph, there exists a valid path to visit all the nodes/vertices.

3. Input Format: The first line contains 2 positive integers n and e - the number of vertex, the number of edge. Then the e lines contains all the edges.

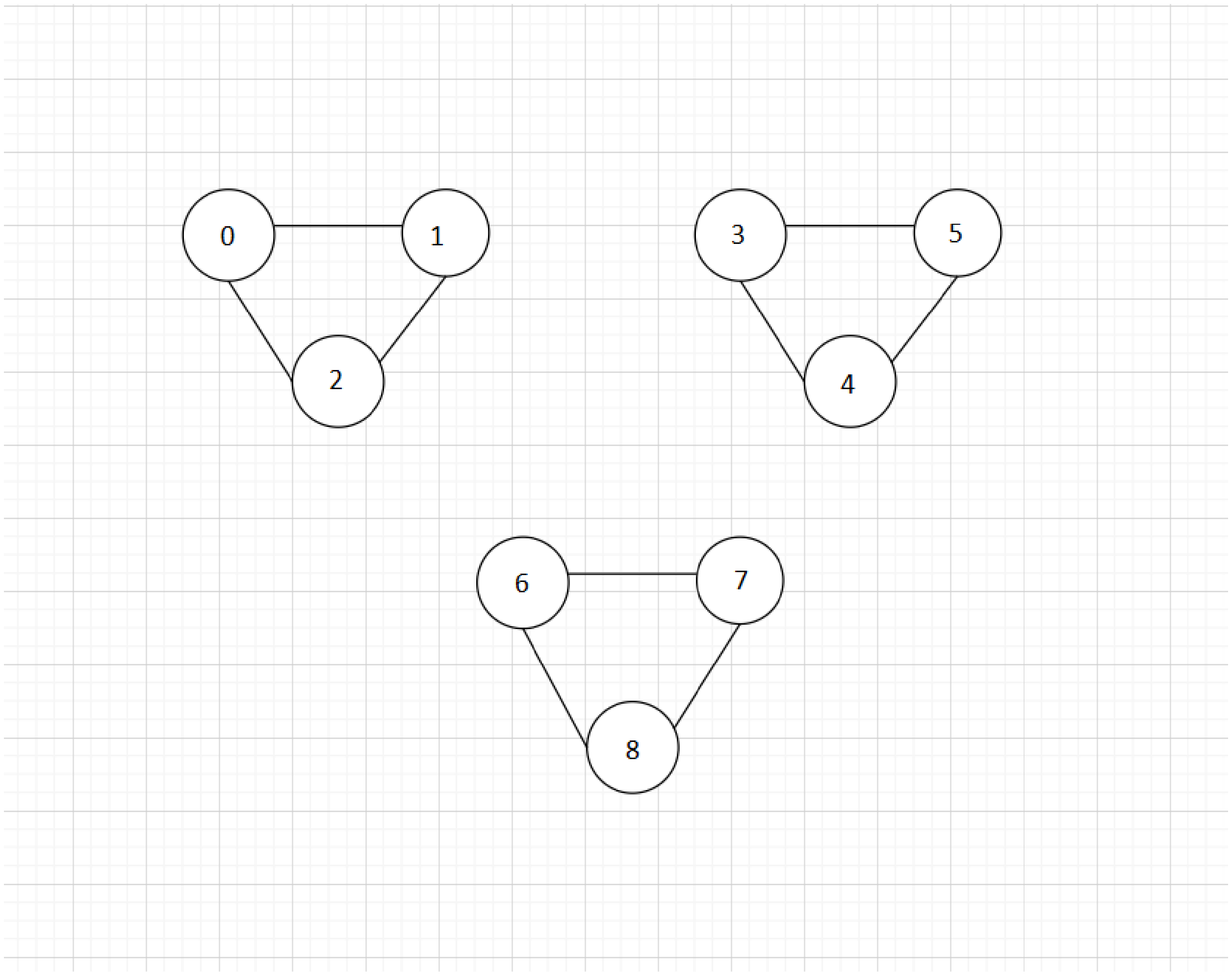
Example:

Input	Output

3 3
0 1
1 2
2 0
	true

Issue Faced: I faced problem while implementing the DFS function. Although the task asks to write a function that returns a Boolean value, I couldn’t implement that feature resulting in constructing a void DFS function instead.

TASK 1(b):



References:

1. https://www.baeldung.com/cs/graph-connected-components

TASK 2:

1.Dijkstra Algorithm: Dijkstra algorithm is a single-source shortest path solving algorithm where it specifically finds the shortest distance between two vertices on a non-negative weighed graph.

Overview: The algorithm follows an array of vertices each of the index of the array indicates if the vertex has been visited or not. The algorithm starts from the source vertex and the directly connected vertex/vertices with the source vertex is marked the corresponding weight. Other non-directly connected vertex/vertices are marked as “Undefined”. The algorithm iteratively selects only the unvisited vertex with the smallest weight/distance from the source vertex. It then marks the vertex as ‘visited’ and moves to the shortest weighed vertex and updates their tentative distances if a shorter path exists. This iteration keeps going until the destination vertex is reached or all the vertices are already visited.

Use cases: This algorithm is specifically used to find the shortest paths between nodes in a non-negative weighted graph. There are many real world use cases where this algorithm is used. Like: Google map’s digital mapping, Telephone network, IP routing, Robotic Path, etc.

Advantage and Disadvantages: The major advantage of this algorithm is the time complexity. It has almost linear time complexity where the time complexity only depends on the summation of the number of edges and nodes. Also it guarantees the shortest path from the source vertex to all other vertices only when the edges have non-negative weights. But it also has some drawbacks, one is very apparent which is- this algorithm can’t handle negative weighted edges. Also when number of nodes in a network becomes very large, it gets complicated for the algorithm to efficiently iterate through.

Sample Code Snippet(Pseudocode):

Pseudocode for Dijkstra Algorith:

```
Dijktra(Graph, Source):
 creating vertex set S

 for each vertex v in graph:
         dist[v] = $\infty$
         add v to S
         dist[source] = 0

     while S is not empty
         u = extract-min[S]
         for each neighbour v of u
             relax(u, v)
```

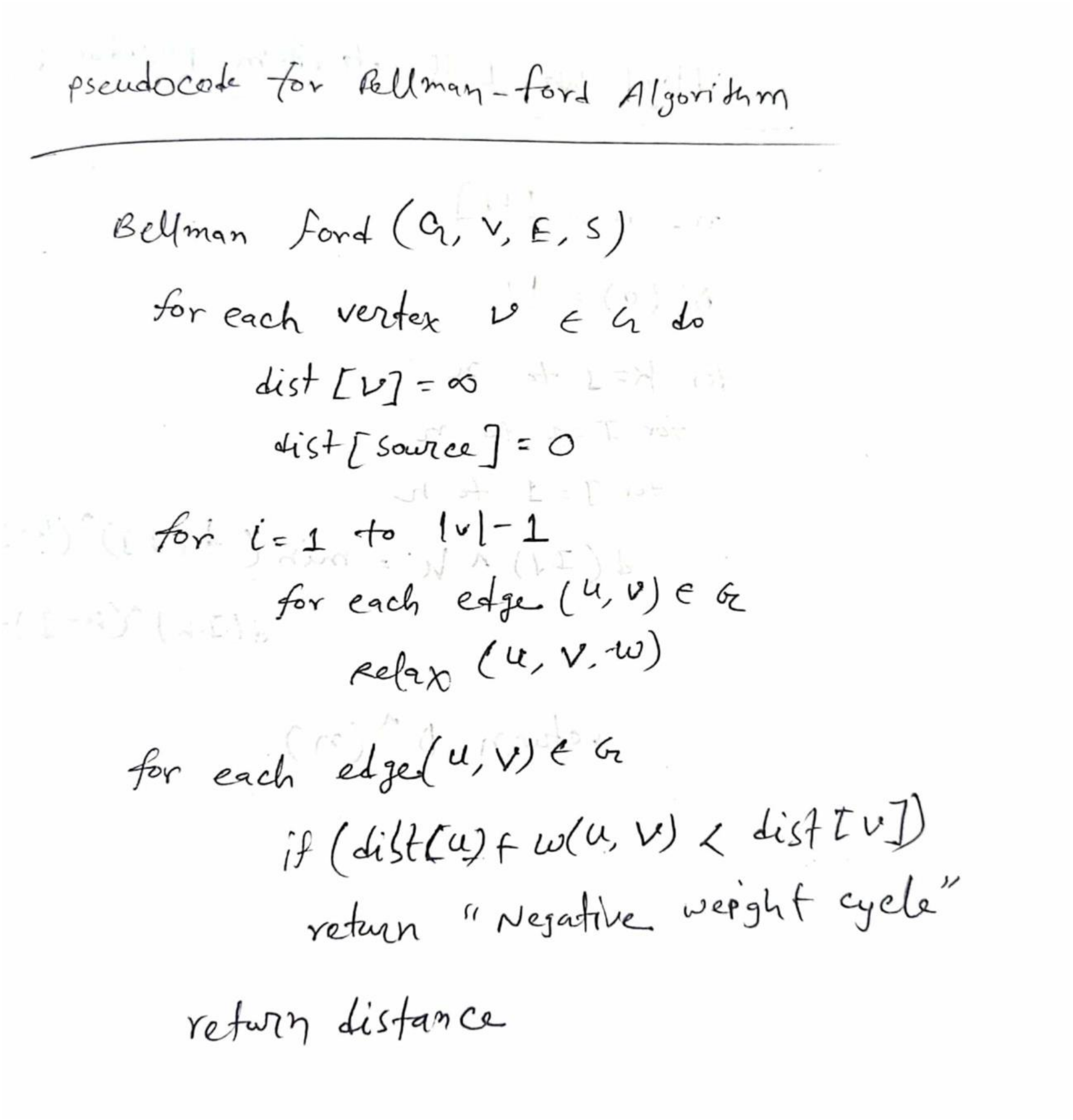
<u>2.Bellman Ford Algorithm:</u> The Bellman Ford algorithm is a dynamic programming-based algorithm that computes the shortest paths from a single source vertex to all other both positive and negative weighted edges.

<u>Overview:</u> This algorithm can follow when there is negative weighted edge and it also can detect negative weight cycle. This algorithm initiates from source node by marking all the vertices undefined value while having the source node as 0 value. Then the algorithm iterates through all the nodes considering that it might have a shorted distance until it finally finds the shortest distance out of the all other vertices. By repeatedly doing this iterative process, the algorithm finally guarantees an optimized result.

<u>Use cases</u>: This algorithm is used mainly when the Dijkstra algorithm can't be used because of the existence of negative weighted edges in the graph. It is used to find the shortest distance from the single vertex to all the other vertices of a weighted graph. The idea of this algorithm is used in the distance-vector routing protocol and this protocol decides how to route packets of data in a network. Also in chemical reactions, this algorithm is used to calculate the smallest possible heat gain or loss.

<u>Advantage and Disadvantages:</u> As this algorithm can work with negative weighted edge it makes more superior than that of Dijkstra algorithm in the sense of versatility. Also this algo can detect any negative weight cycles. But the most severe drawback of this algo is the time complexity as in the worst case scenario, the time complexity can be $O(N^3)$. Also this algo is not fast responsive enough in some complex network topologies.

Code Snippet(Pseudocode):



pseudocode for Bellman-ford Algorithm

```
Bellman ford (G, V, E, S)
  for each vertex v ∈ G do
      dist [v] = ∞
      dist [source] = 0

  for i = 1 to |v| - 1
      for each edge (u, v) ∈ G
          relax (u, v. w)

  for each edge(u, v) ∈ G
      if (dist(u) + w(u, v) < dist [v])
      return "Negative weight cycle"

  return distance
```

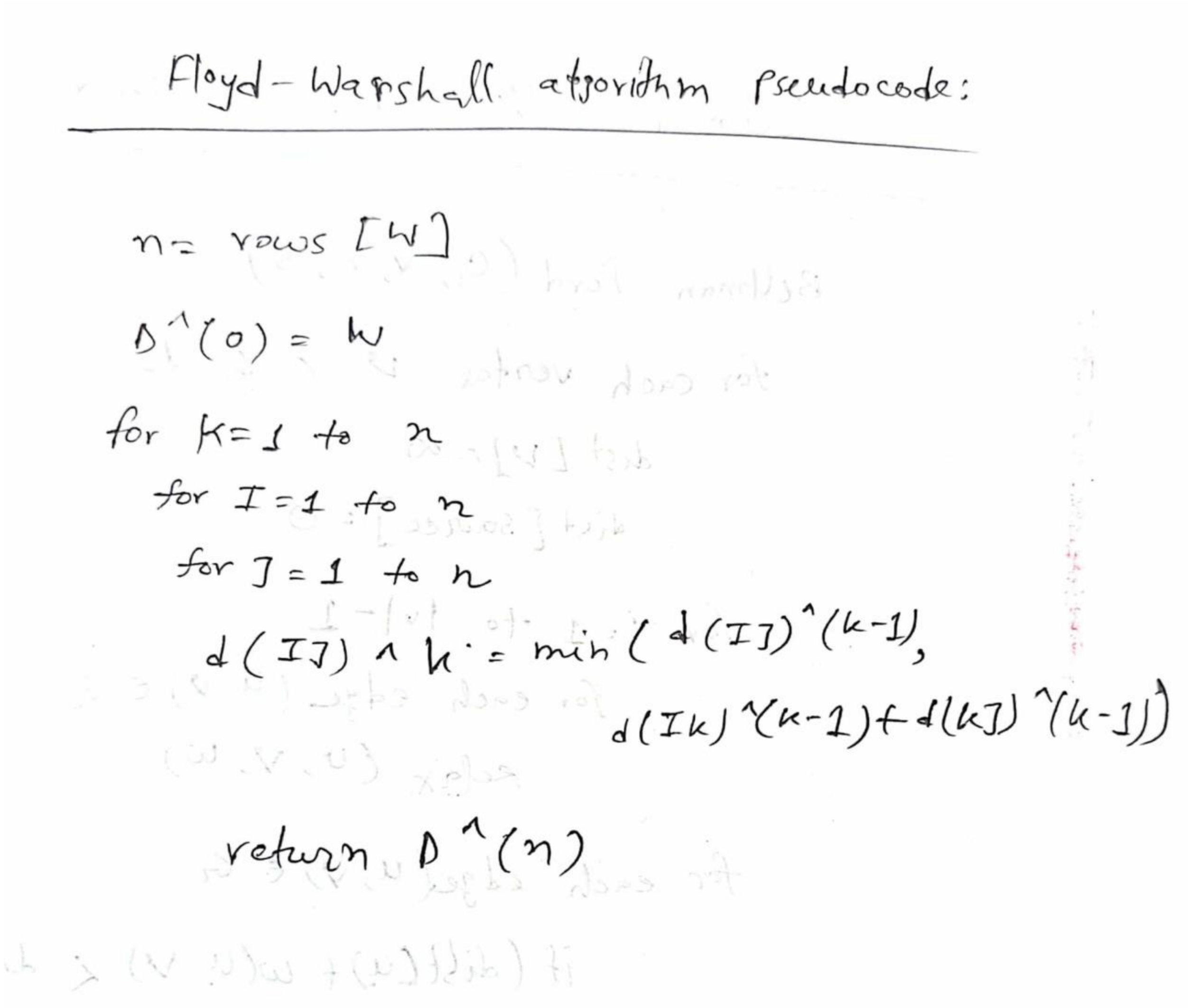
<u>3.Floyd Warshall Algorithm: </u>The Floyd Warshall algorithm is a dynamic programming based algorithm that finds the shortest paths between all pairs of vertices in a weighted graph.

Overview: The algorithm works by initiating a 2D array or an adjacency matrix of size n x n where n is the number of vertices. The distance between the vertices i and j is compared to the distance between I and k + k and j. Here, k is an intermediate matrix which determines how many matrixes are needed to be made. In this iteration, if this distance is found to be shorter, the distance is updated in the matrix. After iterating over all possible intermediate vertices, the algo finally obtains the updated matrix which gives us the shortest possible distance between all the vertices in the graph.

Use cases: This algorithm is mostly used for the dense graphs because the fundamental base of time complexity of this algorithm is the number of vertices in the given graph. This algorithm is mainly used for the fast computation in pathfinder networks and also in the inversion of real matrices. Also if an undirected graph is bipartite or not, this algorithm can be used to check that.

Advantage and Disadvantages: The major advantage of this algorithm is the versatility of solving numerous types of problems. This is superior algorithm than Dijkstra, because it can be implemented in a distributed system. Also it can work with negative edges in a graph. The disadvantages include its slow speed because of the complexity of the algorithm. The time complexity of the algorithm is $O(N^3)$.

Code Snippet(Pseudocode):



Floyd-Warshall algorithm Pseudocode:

```
n= rows [W]

D^(0) = W

for K=1 to n
  for I=1 to n
   for J = 1 to n
      d(IJ)^k = min(d(IJ)^(k-1),
                    d(Ik)^(k-1)+d(kJ)^(k-1))

return D^(n)
```

4.A* Algorithm: A* algorithm is a widely used path finding algorithm which can efficiently find the shortest path from one point to another point in a weighted graph.

Overview: This algorithm works by initiating each step using 2 sets where the open set only contains source node and the closed set remains empty. If,

$$f = g + h;$$

[g = the movement cost to move from the starting point to a given square on the grid, h = the estimated movement cost to move from that given square on the grid to the final destination]

this algorithm in each step picks a node according to the lowest f value, and processes that cell, which is often called heuristic approach.

Usage: This algorithm is used in many video games such as finding enemy in a given parameter from goal. This is often referred as common pathfinding problem. This algorithm can also be used in GPS system, to find shortest paths between a pair of given co-ordinate. This algorithm can also be used in 3D grid, just like as implemented in 2D grid.

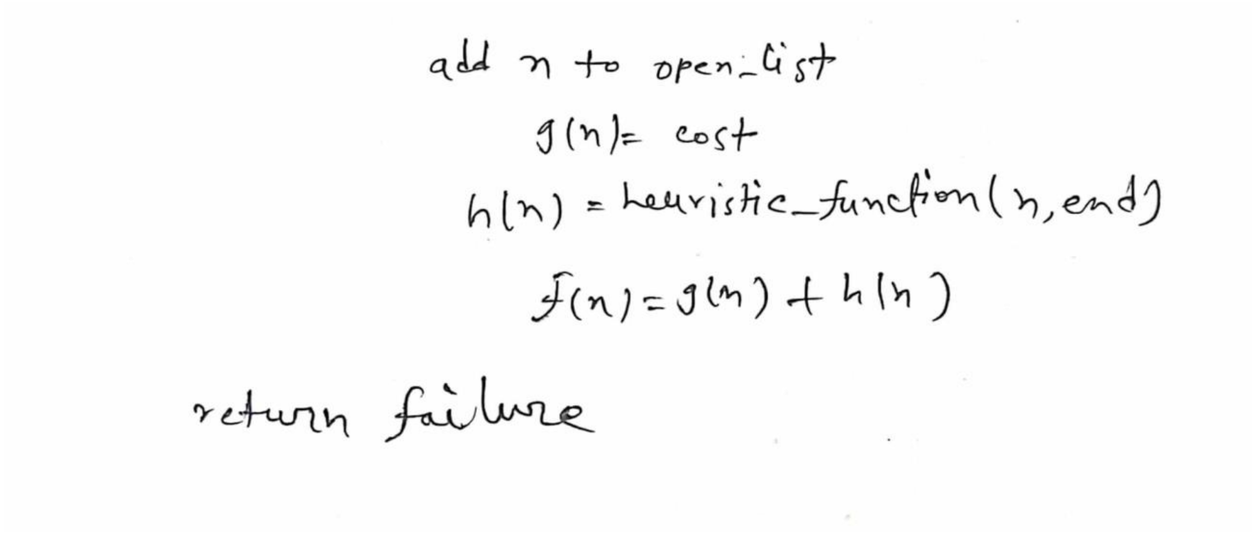
Advantages and Disadvantages: It is one of the best heuristic search techniques and used in solving complex search problems. Also it is quite versatile algorithm which can be used in wide range of problems. It is basically an extended version of Dijkstra Algorithm. But, this algorithm also have some drawbacks, one of which is the complexity problem. Also this algorithm has the tendency to ignore long job because of its memory requirements as it keeps all the generated nodes in the memory.

Code Snippet(Pseudocode):

A* search atgorithm pseudocode

```
Initialize
          Open list = {start}
          closed_list = { }
          g(start) = 0
          h(start) = heuristic_function(start, end)
          f(start) = g(start) + h(start)

While open_list is not emety
          m = Node on top of open_list, with least f
     if m == end
          return
     remove m from open_list
     add m to closed_list
     for each n in child(m)
               if n in closed_list
                         continue
          cost = g(m) + distance(m, n)
          if n in open_list and cost < g(n)
                    remove n from open_list as new path
                                                   is bette
      if n in closed_list and cost < g(n)
                    remove n from closed_list
      if n not in open_list and n not in closed_list
```



```
            add n to open_list
                g(n)= cost
            h(n) = heuristic_function(n, end)
                f(n) = g(n) + h(n)
    return failure
```

References:

1. https://www.geeksforgeeks.org/applications-of-dijkstras-shortest-path-algorithm/
2. https://www.geeksforgeeks.org/introduction-to-dijkstras-shortest-path-algorithm/
3. https://brilliant.org/wiki/bellman-ford-algorithm/#:~:text=A%20version%20of%20Bellman%2DFord,through%20to%20reach%20its%20destination.
4. https://history-computer.com/understanding-the-floyd-warshall-algorithm-with examples/#:~:text=How%20does%20the%20Floyd%2DWarshall,k%20is%20an%20intermediate%20vertex.
5. https://www.scaler.com/topics/data-structures/floyd-warshall-algorithm/
6. https://www.geeksforgeeks.org/a-search-algorithm/

Microcontroller Part

TASK 1:

Circuit:
https://www.tinkercad.com/things/lDRIgqeggIj?sharecode=XVUf4sENG47ouX1TWrZqTG64LJQwMMLMGY3agPsbw5A

Master circuit code: https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_rumman_adib_task_011.ino

Slave circuit code: https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_rumman_adib_task_012.ino

Description: The circuit implements a connection from one microcontroller to another microcontroller (2 arduino uno) in I2C protocol. The A4, A5, GND pins of one Arduino Uno are connected with another Aduino Uno to the corresponding A4, A5, GND pins. Thus the I2C connection is accomplished. The “Wire.h” library helps to establish the I2C connection from the Master to Slave by initializing “Wire.Begin()”. With the Slave Address of 8, the master sends a predefined message “Project Altair” to the slave. The transmission of this message begins by the “Wire.write()” function and ends with “Wire.endTransmission()” function. Then the slave iterates in the loop function and access the transmitted string character by character with “Wire.available()” function and reads with “Wire.read()” function. Then it prints the message on the serial monitor with “Serial.print()” function.

Problem Faced: To transmit the predefined message from master to slave, the message was initially stored in a string, but it was throwing an error. Later the string variable was converted into character array variable and it worked. Because the function “Wire.read()” reads the message character by character.

References:

1. https://electropeak.com/learn/connect-two-arduino-boards-using-i2c-communication-protocol/
2. https://forum.arduino.cc/t/write-serial-monitor-text-to-i2c/354657
3. https://www.ee-diary.com/2022/03/i2c-communication-between-arduino.html

TASK 2:

Circuit:

https://www.tinkercad.com/things/1ZNqy9uHm2L?sharecode=CdwRyL-uMR9efagcT0MtVOt9aHx5cYnmm3OTQSDrSHI

Code: https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_rumman_adib_task_021.ino

Description: The circuit implements the basic control of a DC motor using an Arduino Uno and an encoder. The calculated RPM of the motor is then printed on the serial monitor. The code calculated the RPM by measuring the position of the encoder and the required time of each revolution. The “Encoder.h” library files are Included to work with the encoder. Then the serial communication is established using necessary functions. The RPM is calculated using a user defined function where it takes the number of rotation and the required time for each revolution in milliseconds as two parameters. The it calculates the RPM using the formula (rotation/time)*60 and returns the integer value of the RPM. In the output pins, two PWM signals of different magnitude are applied on two output pins. These are 30 Hz, 0 Hz. The calculated RPM is then showed on the serial monitor with a delay of 1 second.

Problem Faced: The problem arose at the calculation of RPM. The required time for each revolution was passed in milliseconds. Initially the RPM was calculated using the millisecond unit, thus resulting in erroneous output. Then the time was converted in seconds thus the correct output was obtained.

References:

1. https://www.programmingboss.com/2022/03/control-dc-motor-with-rotary-encoder.html#gsc.tab=0
2. https://www.electroniclinic.com/arduino-dc-motor-speed-control-with-encoder-arduino-dc-motor-encoder/

THE END