**Project Altair Recruitment**

**Name: Rumman Adib**

**Student ID: 210041131**

**Department: CSE**

**……………………………………………………………………………………….**

**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
7. Stepper:

…………………………………………………………………………………….

**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.

**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>.

**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](https://en.m.wikipedia.org/wiki/H-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.

**References:**

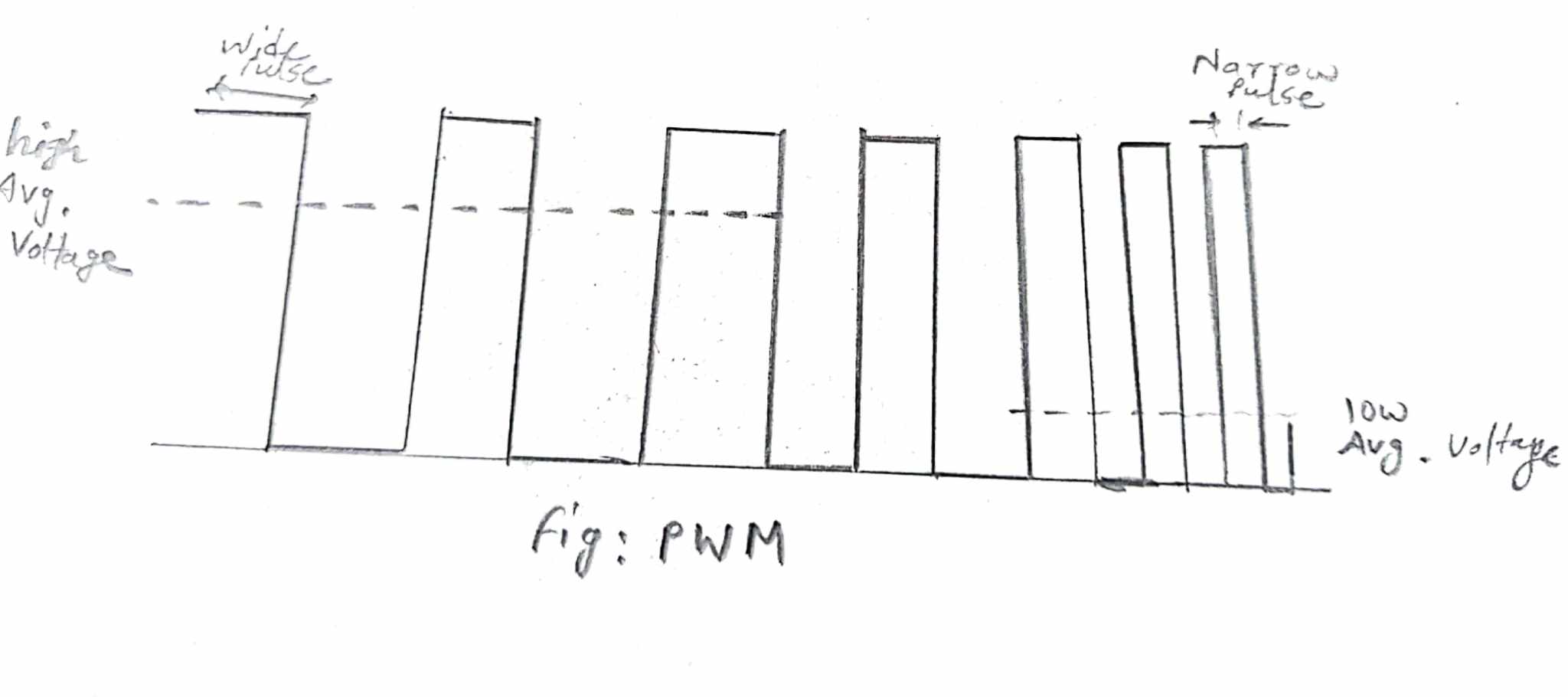
1. [https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control- tutorial-l298n-pwm-h-bridge/#:~:text=L298N%20Driver,and%20explain%20how%20it%20works](https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-%20%20tutorial-l298n-pwm-h-bridge/#:~:text=L298N%20Driver,and%20explain%20how%20it%20works).
2. <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>.
3. <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>.
4. <https://www.wellpcb.com/IR2104.html#:~:text=The%20IR2104%20IC%20is%20a,shifter%20and%20a%20power%20amplifier>.

**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.



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.

**References:**

1.https://www.digikey.com/en/blog/pulse-width-modulation#:~:text=Pulse%20width%20modulation%20turns%20a,to%20when%20it%20turns%20off.  
2.https://en.wikipedia.org/wiki/Pulse-width\_modulation  
3. <https://www.circuitbasics.com/introduction-to-dc-motors-2/>

**…………………………………………………………….**

**Logical Part**

**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:

**Bellman Ford Algorithm:** 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.

Overview: 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.

Use cases: 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.

Advantage and Disadvantages: 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(N3). Also this algo is not fast responsive enough in some complex network topologies.

Code Snippet:

**Floyd Warshall Algorithm:**

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.

**…………………………………………………………….**

**Microcontroller Part**

**TASK 1:**

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