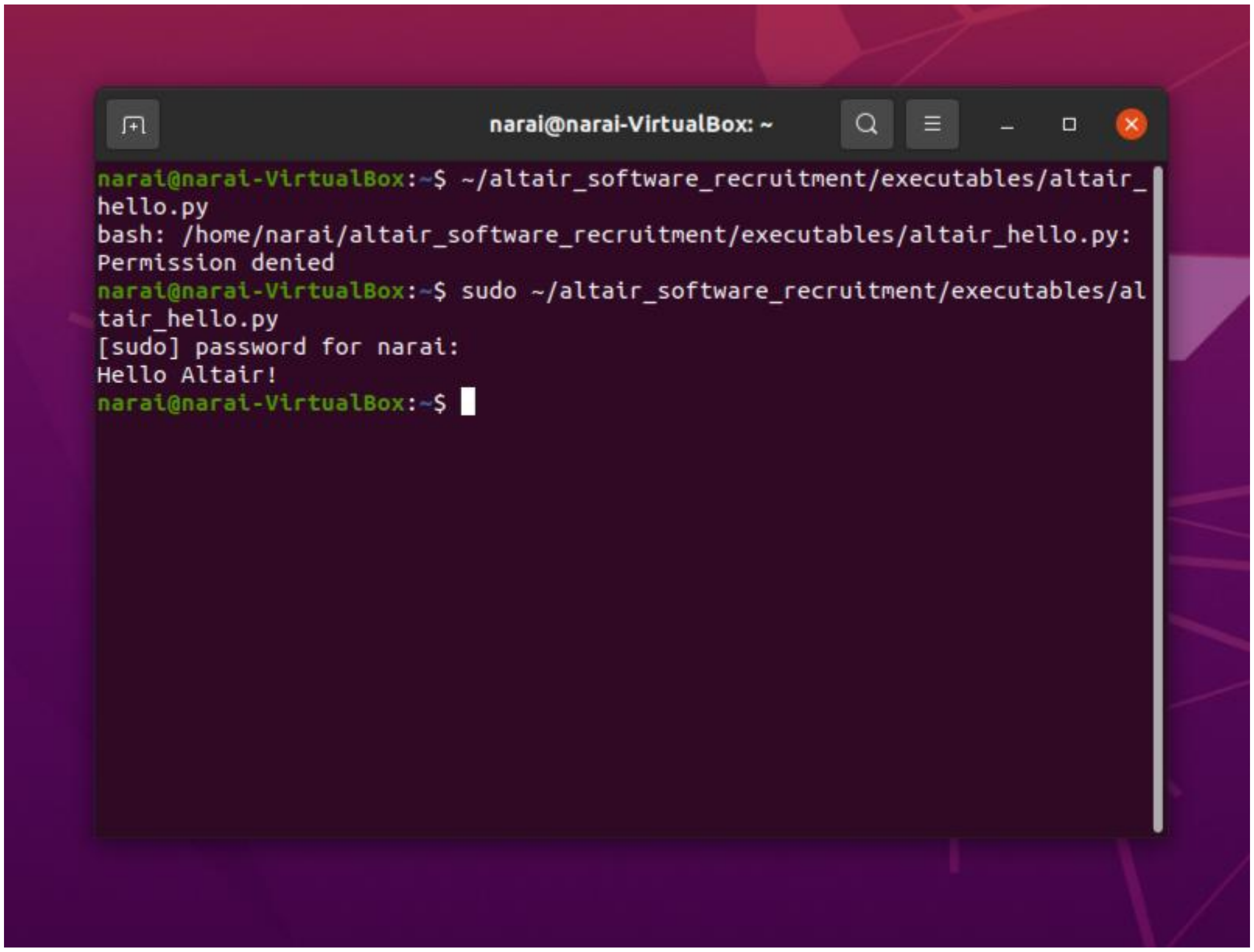
Microcontrollers

Hyperlink –

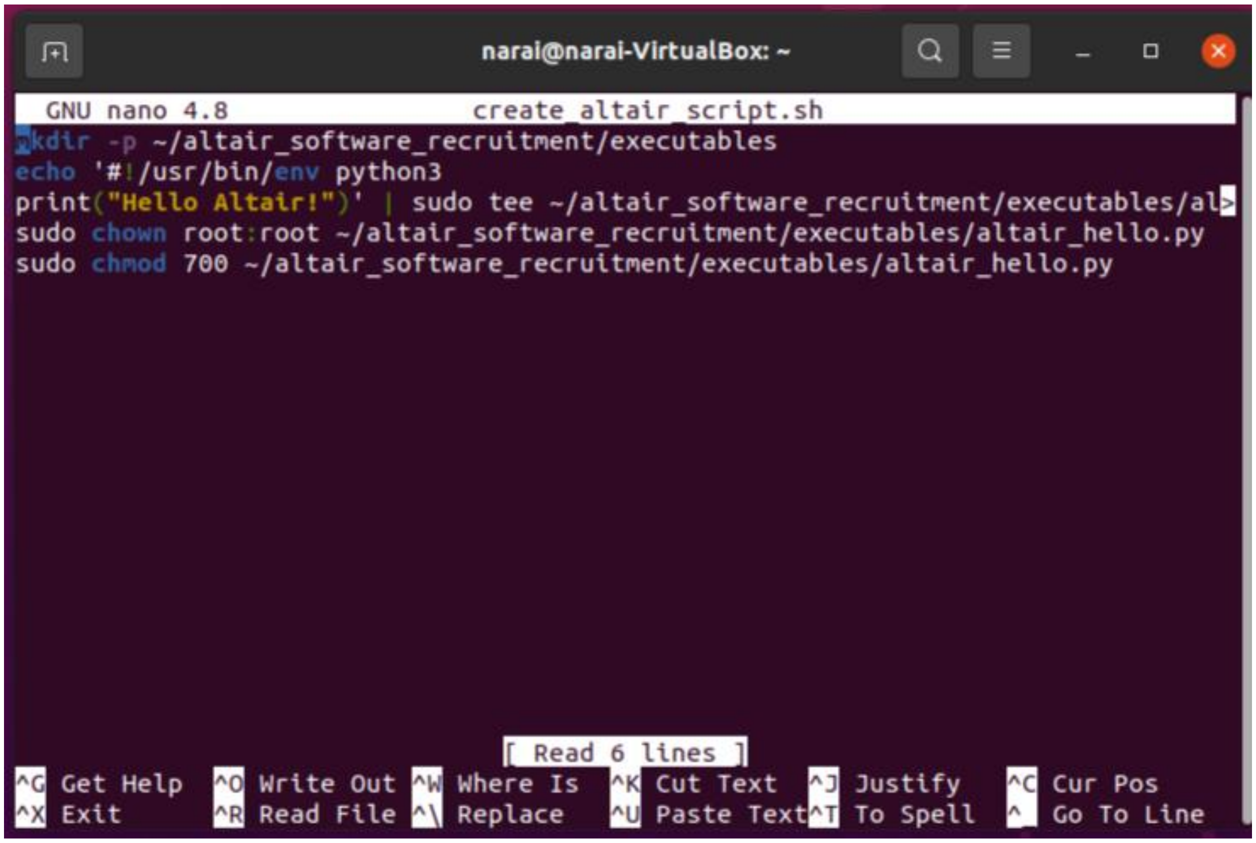
https://www.tinkercad.com/things/8c6mxcGr5Lb-nazifaanjum230041155task1microcontrollers

Unix Terminal



```
narai@narai-VirtualBox:~$ ~/altair_software_recruitment/executables/altair_
hello.py
bash: /home/narai/altair_software_recruitment/executables/altair_hello.py:
Permission denied
narai@narai-VirtualBox:~$ sudo ~/altair_software_recruitment/executables/al
tair_hello.py
[sudo] password for narai:
Hello Altair!
narai@narai-VirtualBox:~$
```

Done in virtual box



```
  GNU nano 4.8                    create_altair_script.sh
mkdir -p ~/altair_software_recruitment/executables
echo '#!/usr/bin/env python3
print("Hello Altair!")' | sudo tee ~/altair_software_recruitment/executables/al>
sudo chown root:root ~/altair_software_recruitment/executables/altair_hello.py
sudo chmod 700 ~/altair_software_recruitment/executables/altair_hello.py







                               [ Read 6 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Sensors (Theoretical)

A. 1. Global Positioning System (GPS):

Satellite-based navigation system that provides location and time information anywhere on earth.

Useful for:

- Long distance navigation
- Outdoor rover localization
- Absolute position reference

2. Inertial Measurement Unit (IMU):

Electronic device that measures orientation, velocity and Gravitational forces using accelerators, gyroscopes and Magnetometers.

Useful for:

- Short term motion tracking
- Slip detection
- Orientation Estimation
- GPS-denied navigation meaning the rover can find its way without GPS

3. 3D depth sensors

   - Depth Sensing Camera
     Camera that measures distance to objects for each pixel, creating depth maps.
     Useful for:
       - Object recognition
       - 3D mapping
       - Obstacle avoidance
   - LIDAR
     Laser based system that measures distance by timing light pulse returns.
     Useful for:
       - High precision mapping
       - Long range obstacle detection

- SLAM(Simultaneous localization and mapping)
- Terrain Analysis

4. Wheel Odometry:

Estimating position by counting wheel rotations and wheel geometry

Useful for:

- Relative position tracking
- Short distance precision
- Backup when other sensors fail
- Continuous motion estimation

B. Musafir 3.0 design

Component model names and their price:

1. GPS- Ublox NEO-M8N

Price: 3.5k-4.5k BDT Source : RoboticsBD

2. IMU- MPU6050

Price: 500-700 BDT Source : RoboticsBD/techshopbd

Depth sensing and perception

3. LIDAR- RPLIDAR A1

Price: 13k-20k BDT Source : RoboticsBD

4. Camera- Raspberry Pi Camera Module V2

Price: 5k-9k BDT Source : RoboticsBD

Processing

5. Main Computer- Raspberry Pi ….

Price- 9.5K-12K   Source : RoboticsBD

6. Microcontroller- Arduino Mega 2560

Price-1.8K-2.5K Source : RoboticsBD

7. Odometer- Rotatory Encoder Module

Price: 150-250 BDT Source : Robotics BD

Wireless Communication:

8. LoRa or WiFi

9. Motors and Batteries

10. Wheels and chassis (price- 3K)

11. Software- Operating System: Ubuntu Linux

Robotics Framework- ROS2

And various kinds of sensors

C. LIDAR VS Depth Camera

1. Budget choice:

   LIDAR-(like RPLIDAR-price 18k-20k BDT)

Pros:

- 360 degree full coverage around rover
- Works in bright sunlight
- Precise measurement
- SLAM

Cons:

- Expensive
- Can’t read ARUco tags or recognize objects
- Only measures distance can’t give color/image

Depth Sense Camera-(like intel Realsense- 18k+ BDT)

Pros:

- Can do multiple jobs- Depth+ object detection+ ARUco reading
- Color images for object recognition
- Good for close-range tasks

Cons:

- Expensive (more than LIDAR)
- Needs more computing power
- Doesn’t work properly in Bright sunlight

What I would choose:

Depth-sensing camera

If we choose LIDAR, we won't be able to perform certain tasks that the mission requires, such as ARUco tag scanning and recognizing objects; it can only avoid objects. Even on a tight budget, these are essential tasks that can't possibly be cut back on. However, if there is a severe shortage of funds, LIDAR can be a go-to choice instead of a Depth-sensing camera.

Sensors: Practical part

1. A) The problem is occurring due to floating inputs. Here in the pinMode function, the pins are set to input without pull-up resistors, which enables the resistor without which

encoder picks up electrical noises and gives random readings

1. B) The change should be: in the pinMode function instead of pinMode(pinA, INPUT) and pinMode(pinB,INPUT) , it should be pinMode(pinA, INPUT_PULLUP) and pinMode(pinB, INPUT_PULLUP).

2. PPR (pulse per revolution) is the number of electrical pulses an encoder generates when the wheel makes one complete 360-degree rotation. Without PPR we wouldn’t know the actual distance covered by the rover. So, using basic tools along with the rover, we can determine the PPR through the following ways:
   1. Manually: Firstly, we need the circumference of the wheel of the rover. Using a basic tool like a scale, we can determine the radius of the wheel, then we calculate the circumference using the formula 2 x π x radius. Then, we mark the starting point with a marker, we push the rover forward so that it covers a distance exactly equal to its circumference, we calculate the number of pulses sent by the encoder, and this count is the PPR.
   2. Using code: We can use code to count the number of pulses in one full rotation using Arduino.

3a. Encoders measure how far the rover has traveled. But it doesn't always give an accurate measurement. Encoders generally measure distance based on the number of rotations of the wheels of the rover, but in uneven terrains, it becomes difficult to be precise. For example, if the terrain is slippery, or if the rover is on a steep incline or wet surfaces, the wheels might slip, so they will rotate, but no distance will be covered. There could be obstacles on the terrain for which the wheels could drag without spinning. Without any sensors, any accurate measurement is quite impossible.

3b. Encoders cannot always give a precise measurement of the travelled distance, so to reduce the problem, we can use certain sensors and software as follows:

1. GPS:
    - Verifies actual speed vs wheel-reported speed
    - Provides the actual position of the rover
    - Corrects encoder drift, for example, resetting the accumulated errors over time
2. IMU (Inertial Measurement Unit):
    - It provides short-term accuracy when encoders fail
    - Detects slip by comparing IMU acceleration vs wheel acceleration
    - Accelerator detects actual rover motion vs wheel rotation

3. LIDAR:
    - It detects ground texture movement using camera-based detection.
    - Measures ground speed accurately
    - It isn’t affected by wheel slip.
4. Various kinds of software, like slip detection software, Sensor fusion algorithms, and drift correction, can help in reducing the problems.

Communication (Theoretical)

Comparative analysis of the mentioned onboard communication protocol.

UART (Universal Asynchronous Receiver/Transmitter)

The function of UART is similar to two people talking on walkie-talkies.

Core functions:

- It acts as a bridge between parallel and serial communication
- It converts parallel data to serial data
- It is known as the hardware block that handles serial communication automatically
- It is Asynchronous, that means no clock signal is needed
- It connects only two devices

Physical Connection:

- TX – Used to transmit data out
- RX – It receives data in
- GND - Common reference
- Optional flow control**:** RTS (Request to Send), CTS (Clear to Send)

Practical Rover Use:

- **GPS to Arduino** communication
- **Bluetooth module** connection

Source: https://learn.sparkfun.com/tutorials/serial-communication/all

I^{2}C (Inter-Integrated Circuit):

Core Concept:

- It works as a Bidirectional communication bus for multiple devices
- It has a Master-slave architecture as in- one controller, multiple peripherals
- Only 2 wires needed for the entire network
- Each device has a unique address

Physical Connection & Hardware

Required Wires:

- **SCL (Serial Clock)** It acts as the Timing signal from the master
- **SDA (Serial Data)** - Data transfer line
- **Pull-up resistors** on both lines to VCC
- **Common ground** reference

Practical Rover Use:

- Data can be read from IMU sensor (MPU6050)
- Can connect Multiple environmental sensors on one bus

Source: https://www.ti.com/lit/an/slva704/slva704.pdf

SPI (Serial Peripheral Interface):

Core Concept:

- Synchronous as in it uses a clock signal
- Full-duplex – It can simultaneously send and receive data
- Main-subnode architecture formerly known as master-slave

- It has a High-speed interface - faster than I^{2}C and UART

Physical Connection & Signals

4-Wire SPI Interface:

- SCLK (Serial Clock**)** - Clock signal from main
- MOSI (Main Out Subnode In) - Data from main to subnode
- MISO (Main In Subnode Out) - Data from subnode to main
- CS/SS (Chip Select) - Selects which subnode to communicate with

Practical Rover Use:

- **SD card** data logging
- **LCDs**
- **High-speed ADC** converters

Source: https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html

CAN (Controller Area Network)

Core Concept:

- It is known for its robust automative protocol
- As any device can transmit It is known as multi-master
- It is usually error-resistant along with built-in checking

Practical Rover Use:

- **Motor controller** communication
- **Critical systems**
- **Distributed sensor networks**

B. Telemetry System Selection for Rover

Essential Functions:

- Regular status updates from the rover to base
- Emergency manual control from base to rover
- Long range communication
- Low power to save the rover battery

There are multiple options, for example WiFi, Bluetooth, Radio Frequency, LTE, Long range, but most of them are not reliable in that certain context. I pick Long Range for the following reasons

1. Its range is 2-10km
2. Date rate is low
3. Power is very low
4. It penetrates through obstacles
5. It has less interference
6. It is designed for harsh environments

The reasons for which other options fail are:

WiFi:

Its range is too short and needs high power and infrastructure

Bluetooth:

Has a very short range

LTE:

Needs High power and has no signal in remote areas

Radio Frequency:

It has less range than LoRa and has more interference issues

LoRa meets all the above criteria and can even do emergency control via instant manual takeover. It is reliable and even cost-effective

PRACTICAL PART:

Algorithm of the function readChannel()

1. At first, we initialize all the necessary variables, an integer array to store the data from Serial1 and a string buffer, in order to keep the partial data appended
2. We need to check if there's any new Data from Serial1
3. We read all the data and append them to the buffer
4. We look for '<' and '>' and only proceed if it's complete, that is, it ends with '>'

5. We extract the integer between ‘<’ and ‘>’ and we can convert them to integers and store them in an integer array by splitting them into 8 integers
6. Clear the buffer

Github link: https://github.com/na-rai/230041155_software_recruitment/tree/main