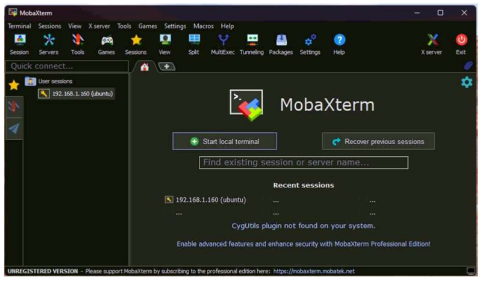
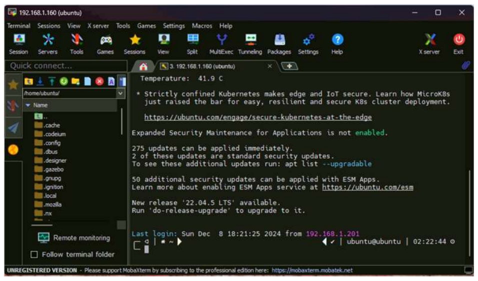
Hiwonder AINEX Robot

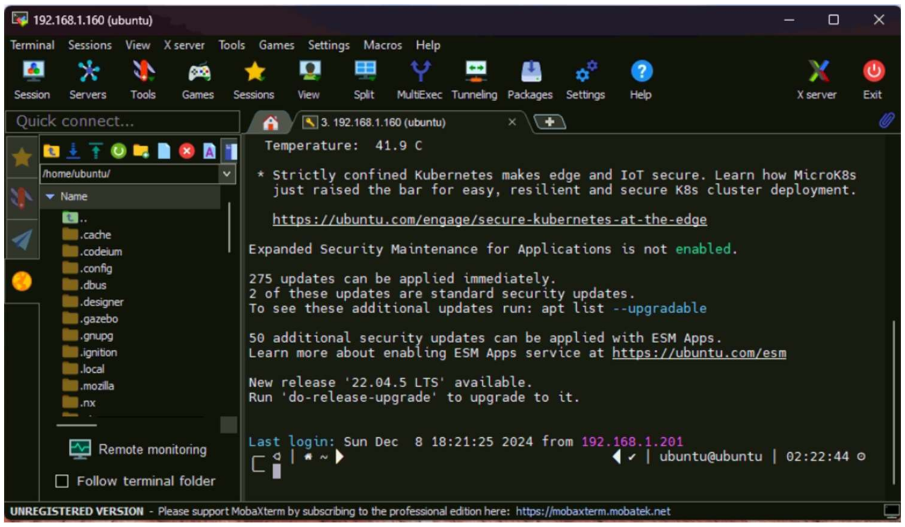
Project: Add custom actions to the robot.

This tutorial assumes that the Robot's Software is already running, and its firmware is already up to the mark. In case of anything like that, we need to first setup the robot's initial firmware. This is given in the next section. To update anything in the raspberry pi, we first need to **connect our CPU with Robot's CPU** (Raspberry Pi 4B+) CPU via **WiFi**. The tutorial for this is given in the file *4. App Installation & Connection*. After connection, we need to install **MobaXterm Software** for communication with the Robot. The installation process is given in the file *Lession 1 MobaXterm Installation & Connection*. Setting up your software is given in the *Lesson 2 PC Software Installation*. MobaXterm is a software like the anydesk. As shown in this image.


MobaXterm
Start local terminal
Recover previous sessions
Find existing session or server name...
Recent sessions
192.168.1.160 (ubuntu)
CygUtils plugin not found on your system.
Enable advanced features and enhance security with MobaXterm Professional Edition!
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net


See on your **left**, here are all the files, you can browse anywhere you want and can open any file. Most files can be open using MobaXterm reader. You can familiarize yourself with this software by going through file *Lesson 1 MobaXterm Installation & Connection*.

After setting up everything, now from **our host computer we can directly work on the robot's computer**. After establishing the connection with the robot, MobaXterm software will show something like this:

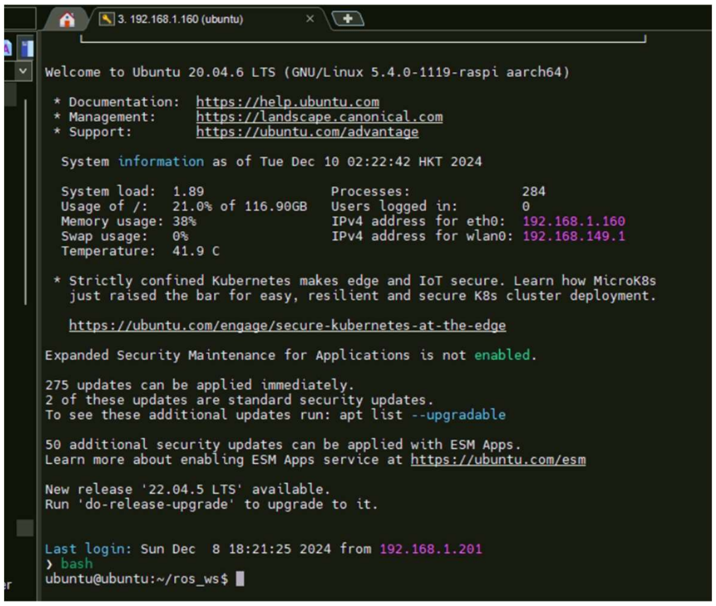

192.168.1.160 (ubuntu)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
3. 192.168.1.160 (ubuntu)
/home/ubuntu/
Name
.cache
.codeium
.config
.dbus
.designer
.gazebo
.gnupg
.ignition
.local
.mozilla
.nx
Remote monitoring
Follow terminal folder

```
  Temperature:  41.9 C

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

275 updates can be applied immediately.
2 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

50 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sun Dec  8 18:21:25 2024 from 192.168.1.201
                                                   ✓ | ubuntu@ubuntu | 02:22:44
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

The terminal is called **Z Shell** terminal. We need our terminal to be **bash** terminal. For that, we will write *bash*

To enter the bash terminal. This will look like this.



3. 192.168.1.160 (ubuntu)

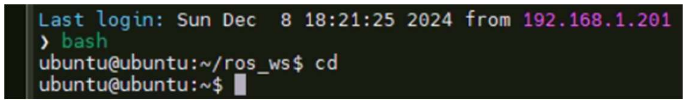
```
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-1119-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Dec 10 02:22:42 HKT 2024

  System load:  1.89               Processes:               284
  Usage of /:   21.0% of 116.90GB  Users logged in:         0
  Memory usage: 38%                IPv4 address for eth0:   192.168.1.160
  Swap usage:   0%                 IPv4 address for wlan0:  192.168.149.1
  Temperature:  41.9 C

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

275 updates can be applied immediately.
2 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

50 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sun Dec  8 18:21:25 2024 from 192.168.1.201
❯ bash
ubuntu@ubuntu:~/ros_ws$ 
```

The image shows that we are in the **ros_ws directory**. This directory is where all our robot's codes and files are stored. To get familiarization with the ubuntu commands please refer to the document named *Linux Commands*

We can control this robot in many ways, and some **action files are already built in.** Before starting anything, its better to get yourself familiarized by the robot and its interface. First write:

cd

this will move you to the home directory.

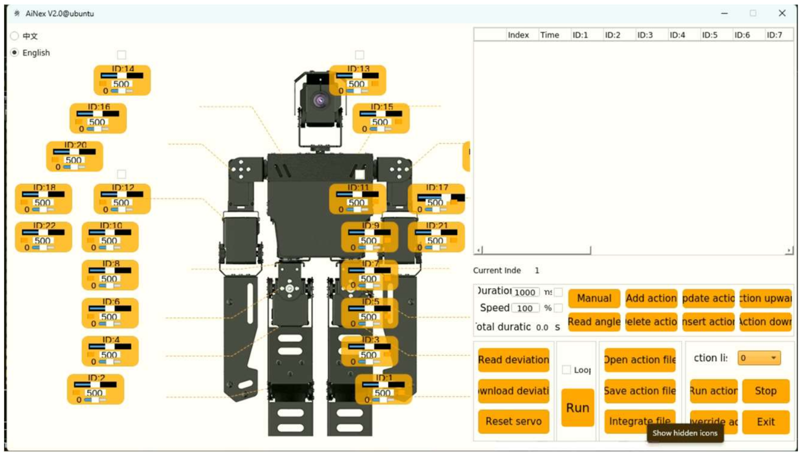


```
Last login: Sun Dec  8 18:21:25 2024 from 192.168.1.201
❯ bash
ubuntu@ubuntu:~/ros_ws$ cd
ubuntu@ubuntu:~$ 
```

Then type:

python3 software/ainex_controller/main.py

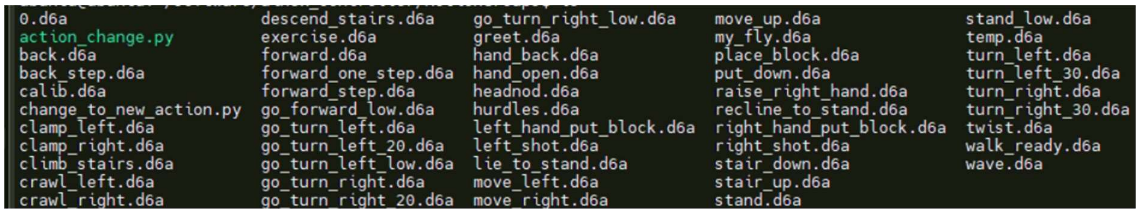
This will open a User Interface; with this you can move any servo motor of the robot.



To get more familiarized by this user interface, please see the file *Lesson 2 PC Software Introduction.* With this user interface, we can do anything with the robot, **move any servo motor** as we want and can reply any motion we want, but keep in mind that the robot is totally dependent on your actions. One wrong move will make the robot fall. The shortcoming of this UI is you cannot control the head of the robot. There are several actions that you can perform with this robot. These actions are stored in the following directory.

ubuntu/software/ainex_controller/Action Files

These functions are in-built and to create our own function, we also need to make a file like this, such that it can be operated from this UI and from the Mobile Application. These files have an extension of **".d6a"**, this is actually a database file.



```
0.d6a                   descend_stairs.d6a    go_turn_right_low.d6a     move_up.d6a                stand_low.d6a
action_change.py        exercise.d6a          greet.d6a                 my_fly.d6a                 temp.d6a
back.d6a                forward.d6a           hand_back.d6a             place_block.d6a            turn_left.d6a
back_step.d6a           forward_one_step.d6a  hand_open.d6a             put_down.d6a               turn_left_30.d6a
calib.d6a               forward_step.d6a      headnod.d6a               raise_right_hand.d6a       turn_right.d6a
change_to_new_action.py go_forward_low.d6a    hurdles.d6a               recline_to_stand.d6a       turn_right_30.d6a
clamp_left.d6a          go_turn_left.d6a      left_hand_put_block.d6a   right_hand_put_block.d6a   twist.d6a
clamp_right.d6a         go_turn_left_20.d6a   left_shot.d6a             right_shot.d6a             walk_ready.d6a
climb_stairs.d6a        go_turn_left_low.d6a  lie_to_stand.d6a          stair_down.d6a             wave.d6a
crawl_left.d6a          go_turn_right.d6a     move_left.d6a             stair_up.d6a
crawl_right.d6a         go_turn_right_20.d6a  move_right.d6a            stand.d6a
```

Now, to create our own custom file for the head nod, we developed a python script and place it in the following location:

ros_ws/src/ainex_tutorial/scripts/serial_servo/headNod.py

To run this file, we need to write the following bash command:

python3 ros_ws/src/ainex_tutorial/scripts/serial_servo/headNod.py

This file will **nod the robot's head twice**. To change anything in the file, we need to open this python file in the **text editor**. The most common text editor for ubuntu is **gedit.**

gedit ros_ws/src/ainex_tutorial/scripts/serial_servo/headNod.py

This will open the headNod script. The explanation to this code is as follows:

```
#!/usr/bin/env python3
# encoding: utf-8
# This script is for AINEX Robot HeadNod. Since, nothing is directly
from the UI
# This script is build by Mr Saad Jameel
# Contact: +92 333 3059002 (WhatsApp Only)
# E-mail: saadjamil1998@gmail.com
```

These are the comments to the code. The first line is necessary, it will direct the compiler for the python executable

```
import time
from ainex_sdk import hiwonder_servo_controller

print('Head-Nod: ')
servo_control                                                     =
hiwonder_servo_controller.HiwonderServoController('/dev/ttyAMA0',
115200)
```

Here, we have first imported the libraries. ***ainex_sdk*** contains the servo controller library. All **24 servos** of this robot are interfaced with a single controller that communicates with the robot via half-duplex UART. For more information read *Lesson 6 Serial Bus Servo Control* file. Then, we initialized the servo controller.

```
def read_servo_position():
    positions = []
    for servo_id in range(1,25):
        pos = servo_control.get_servo_position(servo_id)

        positions.append(pos)
    return positions
```

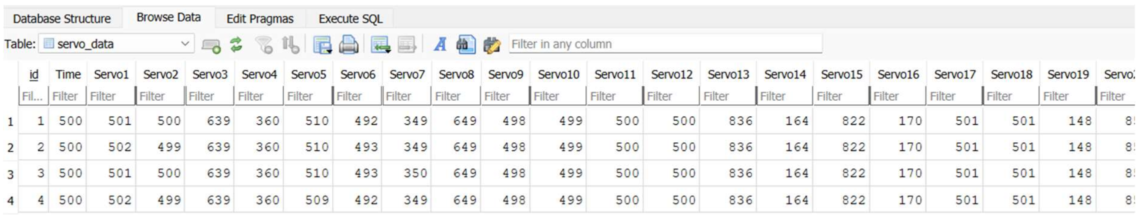
Here, a function is developed to read all the servo positions for final storage in the database.

```
duration = 500
try:
    data = []
    for cycle in range(1, 3):
        for headPos in [600, 500]:
            current_position = read_servo_position()
            # print(current_position)

            # Store data
            record = {"Time": duration}
              record.update({f"Servo{i+1}": positions[i] for i in
range(24)})
            data.append(record)
            servo_control.set_servo_position(23, headPos, duration)
            time.sleep(1)
    print(data)

except KeyboardInterrupt:
    print("Program has Terminated")
```

These lines of codes nod the head of the robot twice. The head position is from **[500, 600].** Here, before the head movement, the robot motor's positions are logged then its head moves from one position to the next and this cycle goes on until **2 head movements**. Finally, the logged values are then stored in a dictionary for exporting it into the database.

```
import sqlite3
from PyQt5.QtSql import QSqlDatabase, QSqlQuery

file_path                                                          =
'/home/ubuntu/software/ainex_controller/ActionGroups/headnod.d6a'

# Function to initialize SQLite database
def init_database():
    global file_path
    conn = sqlite3.connect(file_path)
    cursor = conn.cursor()
    # Create table if it doesn't exist
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS servo_data (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            Time INTEGER,
            Servo1 INTEGER,
            Servo2 INTEGER,
            Servo3 INTEGER,
            Servo4 INTEGER,
            Servo5 INTEGER,
            Servo6 INTEGER,
            Servo7 INTEGER,
            Servo8 INTEGER,
            Servo9 INTEGER,
            Servo10 INTEGER,
            Servo11 INTEGER,
            Servo12 INTEGER,
            Servo13 INTEGER,
            Servo14 INTEGER,
            Servo15 INTEGER,
            Servo16 INTEGER,
            Servo17 INTEGER,
            Servo18 INTEGER,
            Servo19 INTEGER,
                Servo20 INTEGER,
                Servo21 INTEGER,
                Servo22 INTEGER,
                Servo23 INTEGER,
                Servo24 INTEGER
            )
        ''')
    conn.commit()
    conn.close()

# Function to insert data into the database
def insert_data(data):
    global file_path
    conn = sqlite3.connect(file_path)
    cursor = conn.cursor()
    for record in data:
        placeholders = ', '.join(['?'] * 25)  # 1 for Time + 24 for
Servo positions
        values = [record["Time"]] + [record[f"Servo{i+1}"] for i in
range(24)]
        cursor.execute(f"INSERT  INTO  servo_data  (Time,  Servo1,
Servo2,  Servo3,  Servo4,  Servo5,  Servo6,  Servo7,  Servo8,  Servo9,
Servo10,  Servo11,  Servo12,  Servo13,  Servo14,  Servo15,  Servo16,
Servo17,  Servo18,  Servo19,  Servo20,  Servo21,  Servo22,  Servo23,
Servo24) VALUES ({placeholders})", values)
    conn.commit()
    conn.close()

# Initialize database
init_database()

# Insert data into SQLite database
    insert_data(data)
    print("Data has been saved to the database.")
```

This code logs the data into the database with an extension of **".d6a" DB file**. First import the **sqlite3** library and then open a database at the known folder which is given in the file path variable. We chose the same folder where all the action database is stored. The files will look like as a following image.



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: servo_data

	id	Time	Servo1	Servo2	Servo3	Servo4	Servo5	Servo6	Servo7	Servo8	Servo9	Servo10	Servo11	Servo12	Servo13	Servo14	Servo15	Servo16	Servo17	Servo18	Servo19	Servo
	Fil...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	500	501	500	639	360	510	492	349	649	498	499	500	500	836	164	822	170	501	501	148	8
2	2	500	502	499	639	360	510	493	349	649	498	499	500	500	836	164	822	170	501	501	148	8
3	3	500	501	500	639	360	510	493	350	649	498	499	500	500	836	164	822	170	501	501	148	8
4	4	500	502	499	639	360	509	492	349	649	498	499	500	500	836	164	822	170	501	501	148	8