**Prof.** Moran Garabito Carlos Enrique

**Asignatura:** Dinámica y control de robots

EV\_2\_2\_Movimiento de un robot

Sevilla cordero cesar Arturo

Objetivo:

Mover un motor a pasos con un microcontrolador de 32 bits.

Materiales:

Ubuntu con ROS

Microcontrolador (KL25Z)

Driver DV8825A

Motor Nema 23

Cables

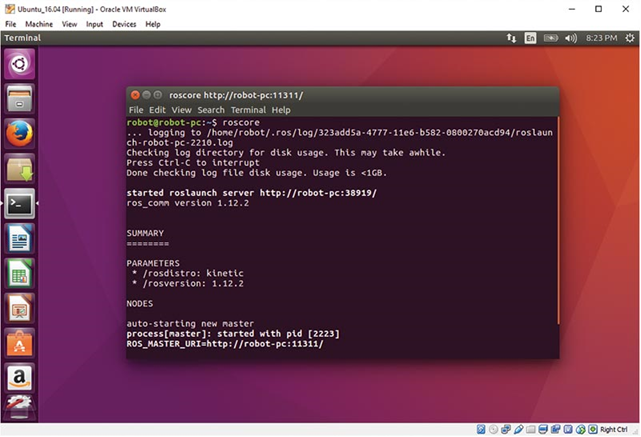
Introducción:

Para la comunicación entre un microcontrolador y ROS es necesario tener instalados e importar (dependiendo el microcontrolador y su forma de programar) la librería rosserial, una vez con estas librerías a continuación se procede a hacer el código para mover un motor a pasos.

Procedimiento:

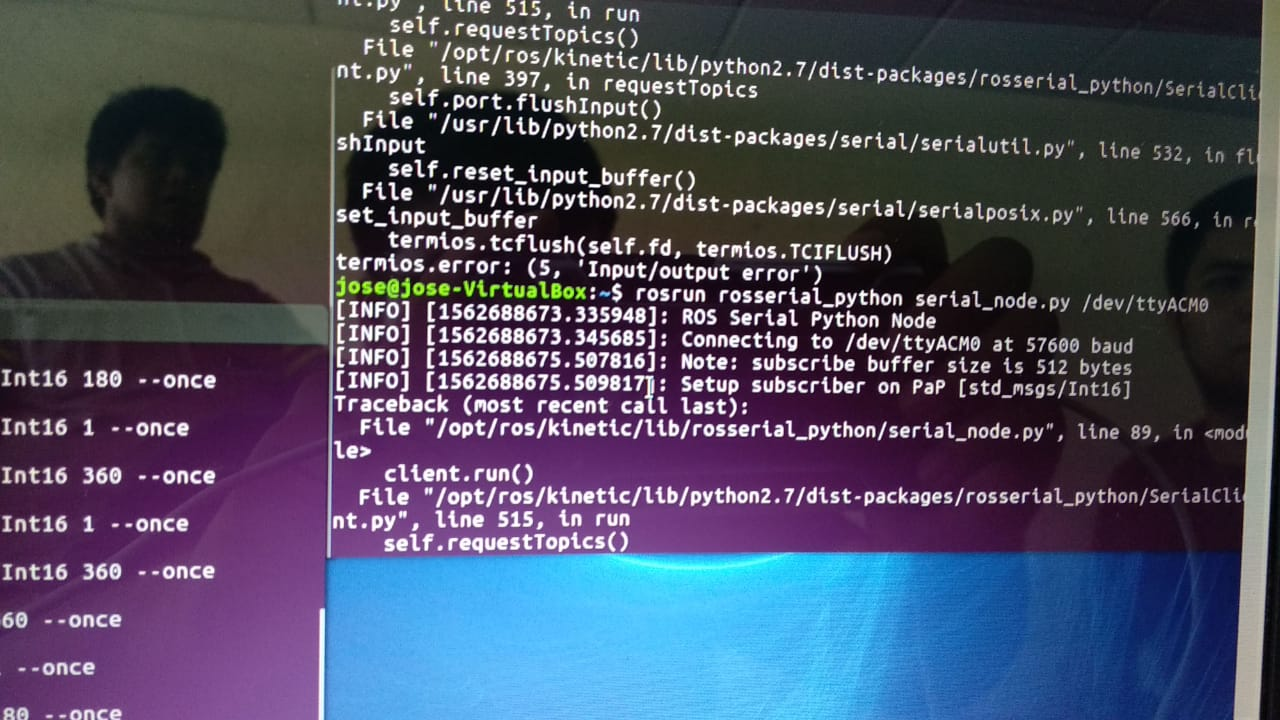
1. Se abre una terminal en la cual se inicia ROS con el siguiente comando

*roscore*



1. Una ves iniciado ROS se abre una nueva terminar en la cual se ingresará el siguiente comando

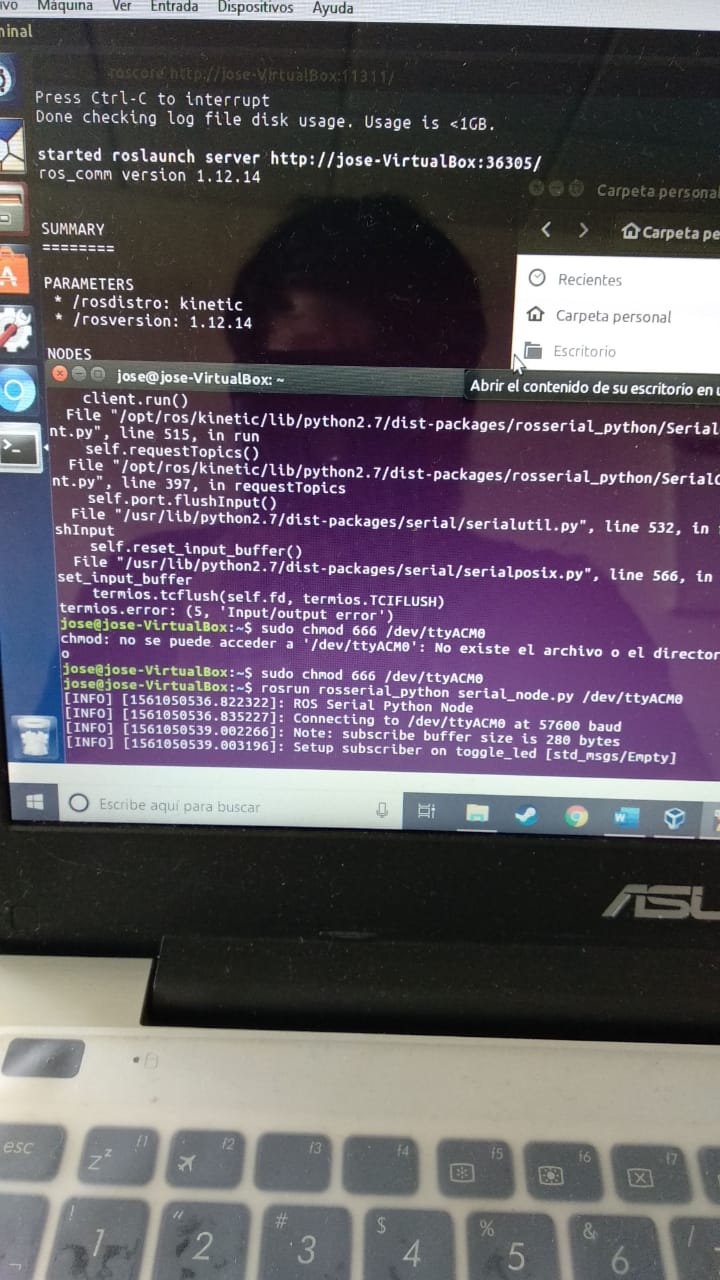
*rosrun rosserial\_python serial\_node.py /dev /ttyAMC0*



Este comando es para conectare el microcontrolador a Ubuntu y así sea reconocido por ROS.

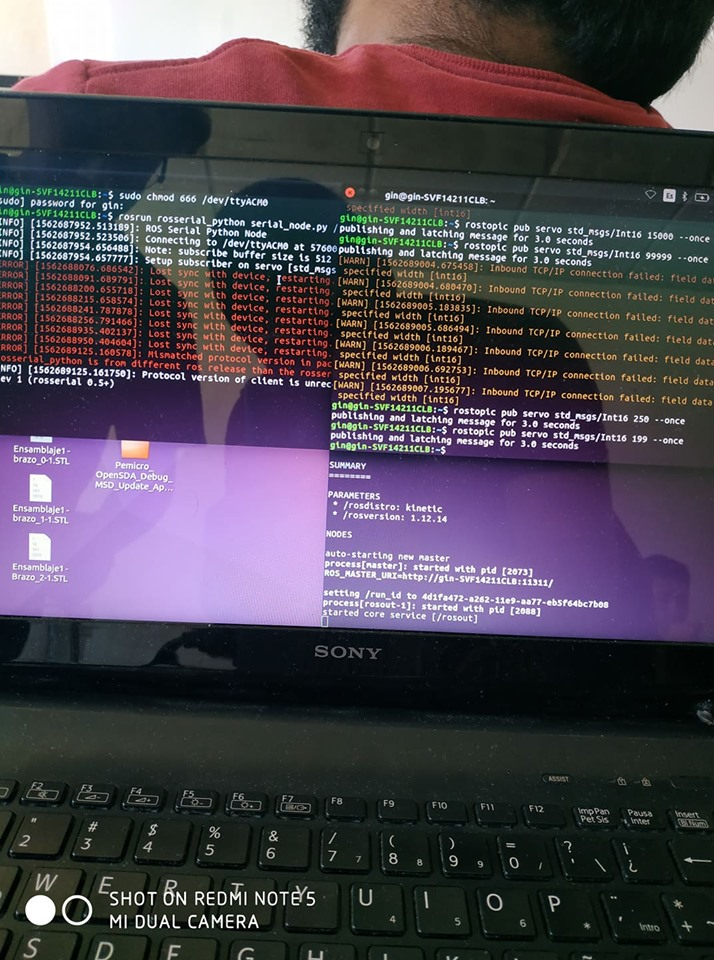
En caso de que el comando de error se debe ingresar el siguiente comando:

*sudo chmod 666 /dev/ttyACM0 or ttyS0*

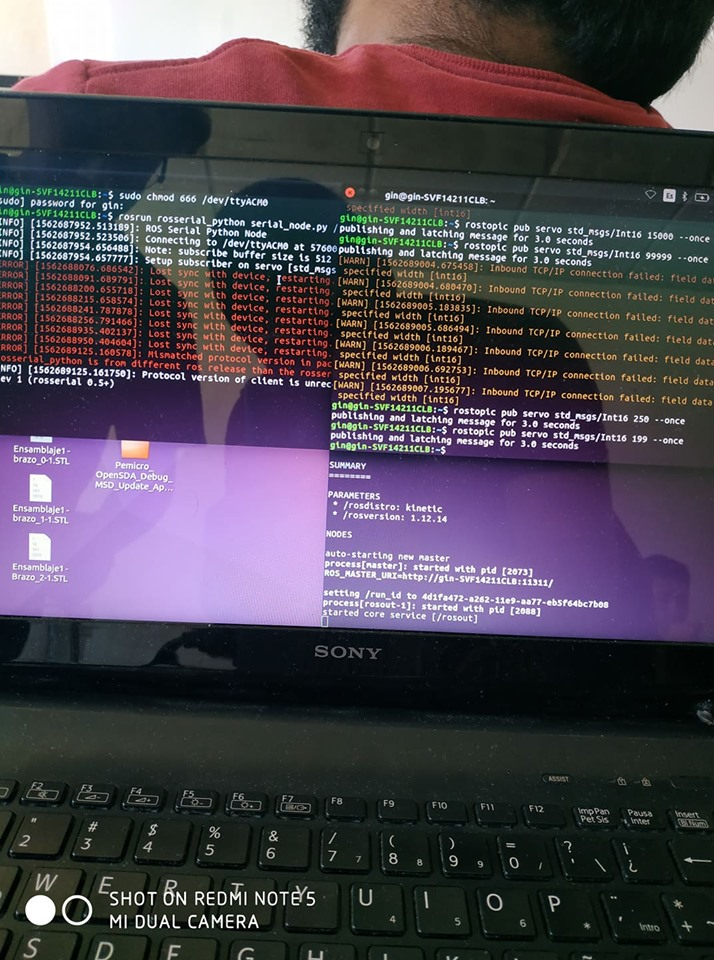


Este comando sirve para que en el puerto en el que este conectado el microcontrolador obtenga los permisos para que se comunique con ROS.

1. Una vez conectado correctamente el microcontrolador a Ubuntu se abre una nueva terminal en la que se ingresara el comando para realizar la comunicación serial entre el micro y ROS, un ejemplo de un comando es el siguiente que se muestra en la imagen.



Resultados:



Comando de comunicación entre ROS y la KL25Z

Movimiento fluido del motor de un lado y otro por medio de los comandos escritos en la terminal de comunicación ros.

Código usado en MBED

#include "mbed.h"

#include <ros.h>

#include <std\_msgs/Int16.h>

ros nh;

DigitalOut step(D2);

DigitalOut dir(D5);

DigitalOut en(D8);

float stepDelay = 0.0016;

void servo\_cb(const std\_msgs::Int16& cmd\_msg) {

) {

int vr = vrec-500;

for (int x = 0; x < vr; x++) {

step=1; dir=0; en=0; wait(stepDelay);

step=0;

}

}

if(vrec < 500) { for (int x = 0; x < vrec; x++) {

step=1; dir=1;

en=0;

wait(stepDelay);

step=0;

}

}

}

ros::Subscriber<std\_msgs::Int16> sub("servo", servo\_cb);

nh.spinOnce();

wait\_ms(1);

}

}

**Con este tipo de movimiento se pudo mover el robot atravez de una comunicación serial de manera efectiva separando el movimiento de las matrices por separado para tener un control uniforme y correcto se a creado una i terface con ross**