



Ingeniería mecatrónica
9°B T/M

Navarro Cervantes José

Prof. Moran Garabito Carlos Enrique

Asignatura: Dinámica y control de robots

EV_2_2_Movimiento de un robot

Objetivo:

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

Materiales:

Ubuntu con ROS

Microcontrolador (KL25Z)

Motor a pasos (con driver)

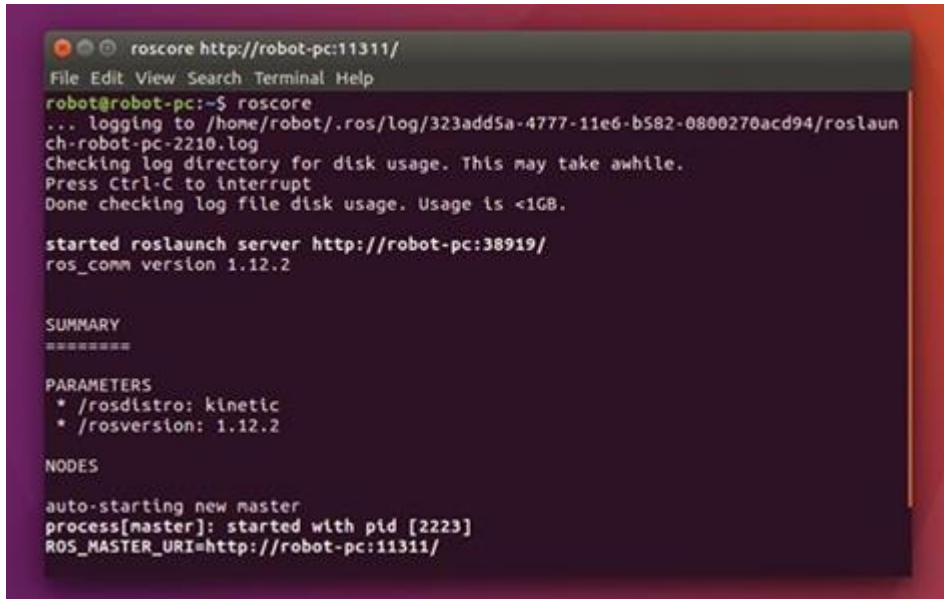
Cables

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



```
roscore http://robot-pc:11311/
File Edit View Search Terminal Help
robot@robot-pc:~$ roscore
... logging to /home/robot/.ros/log/323add5a-4777-11e6-b582-0800270acd94/roslaun
ch-robot-pc-2210.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://robot-pc:38919/
ros_comm version 1.12.2

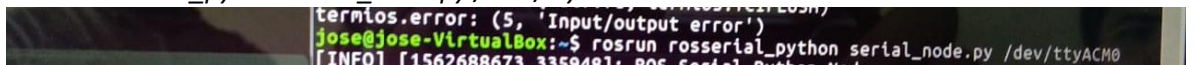
SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.2

NODES
auto-starting new master
process[master]: started with pid [2223]
ROS_MASTER_URI=http://robot-pc:11311/
```

- 2- Una vez iniciado ROS se abre una nueva terminal en la cual se ingresará el siguiente comando

roslaunch rosserial_python serial_node.py /dev /ttyACM0



```
roslaunch rosserial_python serial_node.py /dev /ttyACM0
KeyboardInterrupt
[ERROR] [1562688673.335948]: KeyboardInterrupt
[INFO] [1562688673.335948]: ROS Serial Python Node
```

Este comando es para conectarse 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 ttySO

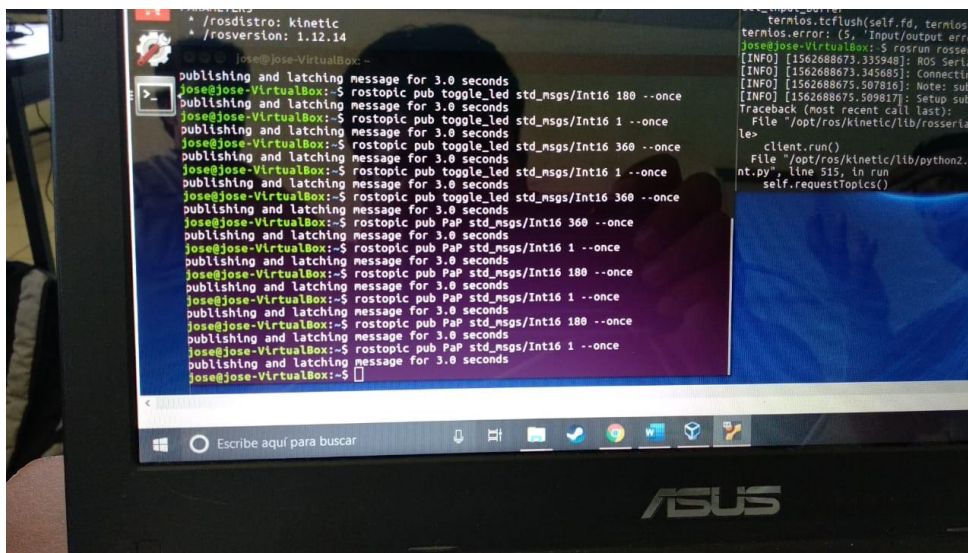
```
chmod: no se puede acceder a '/dev/ttyACM0': No existe el archivo o el directorio
jose@jose-VirtualBox:~$ sudo chmod 666 /dev/ttyACM0
jose@jose-VirtualBox:~$ roslaunch rosserial_python serial_node.py /dev/ttyACM0
[INFO] [1561050536.822322]: ROS Serial Python Node
[INFO] [1561050536.835227]: Connecting to /dev/ttyACM0
```

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

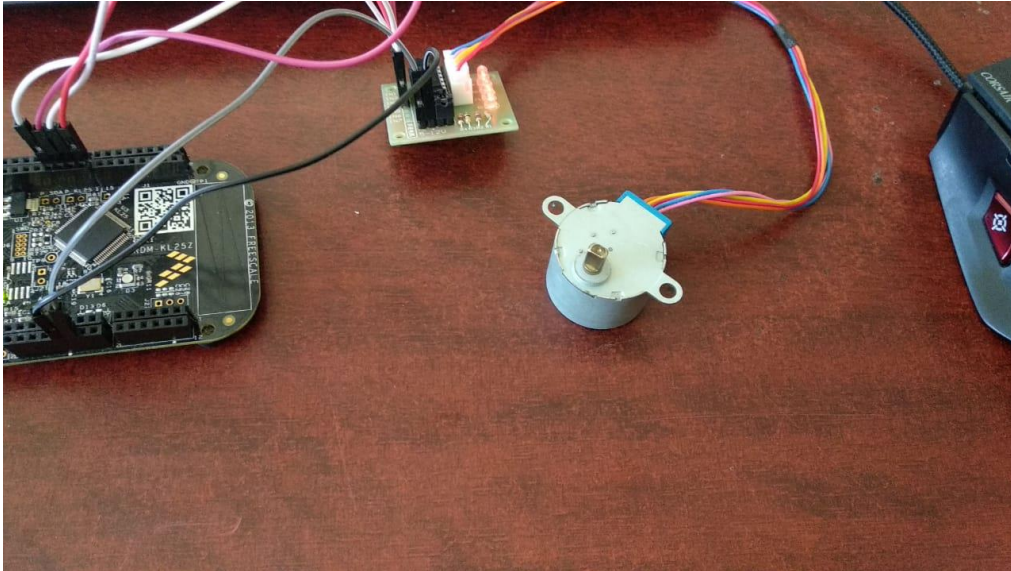
- 3- Una vez conectado correctamente el microcontrolador a Ubuntu se abre una nueva terminal en la que se ingresara el comando para realizar la comunicación serial entre el micro y ROS, un ejemplo de un comando es el siguiente que se muestra en la imagen.

```
publishing and latching message for 3.0 seconds
jose@jose-VirtualBox:~$ rostopic pub toggle_led std_msgs/Int16 180 --once
publishing and latching message for 3.0 seconds
jose@jose-VirtualBox:~$ rostopic pub toggle_led std_msgs/Int16 1 --once
publishing and latching message for 3.0 seconds
jose@jose-VirtualBox:~$ rostopic pub toggle_led std_msgs/Int16 360 --once
publishing and latching message for 3.0 seconds
```

Resultados:



Comando de comunicación entre ROS y la KL25Z



Moto a pasos ya funcionando mediante grados moviéndose en ambas direcciones.

Código usado en MBED

```
#include "mbed.h"
#include <ros.h>
#include <std_msgs/Int16.h>

ros::NodeHandle nh;
DigitalOut IN1(D8);
DigitalOut IN2 (D9);
DigitalOut IN3 (D10);
DigitalOut IN4 (D11);

int16_t step = 0;
int16_t data = 0;
int16_t paso = 0;

void messageCb(const std_msgs::Int16& step_msg){
    step = step_msg.data;
}

ros::Subscriber<std_msgs::Int16> sub("PaP", &messageCb);

int main() {
    nh.initNode();
    nh.subscribe(sub);

    while (1) {
        nh.spinOnce();
        wait_ms(1);
    }
}
```

```

    if (step > 0) {
        data = step;
        data = (data*1.4222222222);
    }
if(data > paso){
    IN1= 1;
    IN2 =1;
    IN3 =0;
    IN4 =0;
    wait(0.01);

    IN1= 0;
    IN2 =1;
    IN3 =1;
    IN4 =0;
    wait(0.01);

    IN1=0;
    IN2=0;
    IN3=1;
    IN4=1;
    wait(0.01);

    IN1=1;
    IN2=0;
    IN3=0;
    IN4=1;
    wait(0.01);
    paso++;
}
if (data < paso) {
//paso1
    IN1=1;
    IN2=0;
    IN3=0;
    IN4=1;
    wait(0.01);
    //paso2
    IN1=0;
    IN2=0;
    IN3=1;
    IN4=1;
    wait(0.01);
    //paso3
    IN1=0;

```

```
IN2=1;
IN3=1;
IN4=0;
wait(0.01);
//paso4
IN1=1;
IN2=1;
IN3=0;
IN4=0;
wait(0.01);
paso--;
}
}
}
```

Conclusión:

Al momento de conectar el motor si este no esta bien conectado en las salidas este no girara como se debe y se quedara trabado en una posición, la forma en la que se ejecuta el programa hecho es que si el valor que se le da en el mensaje el la terminal de ROS es menor al primer mensaje este regresara a la posición ingresada en el primer mensaje.