Ricardo Martínez Jacinto

[Nombre de la empresa]

[Dirección de la compañía]

movimiento de un robot

tarea 4

Después de haber leído el documento proporcionado por el profesor (tarea4) proceder con su realización

Objetivo:

Mover un motor a pasos de acuerdo al numero de grados que se le sea introducido por medio de ROS

Materiales utilizados en esta práctica:

* Arduino y software para programarlo
* Motor a pasos con su respectivo driver
* Jumpers o cables
* Tarjeta Cypress 049-42XX y sofrware para programarlo
* En este caso se trabajará con una máquina virtual con Ubuntu 16

Primero hacemos la programación de las tarjetas

Código en Arduino:

***#include <ros.h>***

***#include <std\_msgs/Int16.h>***

***ros::NodeHandle nh;***

***int pot;***

***int retardo=5; // Tiempo de retardo en milisegundos (Velocidad del Motor)***

***int dato\_rx; // valor recibido en grados***

***int numero\_pasos = 0; // Valor en grados donde se encuentra el motor***

***String leeCadena; // Almacena la cadena de datos recibida***

***void potencia( const std\_msgs::Int16& cifra){***

***::pot = cifra.data;***

***}***

***ros::Subscriber<std\_msgs::Int16> sub("cifra", &potencia );***

***std\_msgs::Int16 res;***

***void setup() {***

***nh.initNode();***

***nh.subscribe(sub);***

***pinMode(11, OUTPUT); // Pin 11 conectar a IN4***

***pinMode(10, OUTPUT); // Pin 10 conectar a IN3***

***pinMode(9, OUTPUT); // Pin 9 conectar a IN2***

***pinMode(8, OUTPUT); // Pin 8 conectar a IN1***

***}***

***void loop() {***

***res.data = ::pot;***

***if (res.data>0){***

***res.data = (res.data \* 1.4222222222); // Ajuste de 512 vueltas a los 360 grados***

***}***

***while (res.data>numero\_pasos){ // Girohacia la izquierda en grados***

***paso\_izq();***

***numero\_pasos = numero\_pasos + 1;***

***}***

***while (res.data<numero\_pasos){ // Giro hacia la derecha en grados***

***paso\_der();***

***numero\_pasos = numero\_pasos -1;***

***}***

***leeCadena = ""; // Inicializamos la cadena de caracteres recibidos***

***apagado(); // Apagado del Motor para que no se caliente***

***nh.spinOnce();***

***delay(1000);***

***} ///////////////////// Fin del Loop ///////////////////////////***

***void paso\_der(){ // Pasos a la derecha***

***digitalWrite(11, HIGH);***

***digitalWrite(10, LOW);***

***digitalWrite(9, LOW);***

***digitalWrite(8, HIGH);***

***delay(retardo);***

***digitalWrite(11, LOW);***

***digitalWrite(10, HIGH);***

***digitalWrite(9, LOW);***

***digitalWrite(8, HIGH);***

***delay(retardo);***

***digitalWrite(11, LOW);***

***digitalWrite(10, HIGH);***

***digitalWrite(9, HIGH);***

***digitalWrite(8, LOW);***

***delay(retardo);***

***digitalWrite(11, HIGH);***

***digitalWrite(10, LOW);***

***digitalWrite(9, HIGH);***

***digitalWrite(8, LOW);***

***delay(retardo);***

***}***

***void paso\_izq() { // Pasos a la izquierda***

***digitalWrite(11, LOW);***

***digitalWrite(10, HIGH);***

***digitalWrite(9, HIGH);***

***digitalWrite(8, LOW);***

***delay(retardo);***

***digitalWrite(11, LOW);***

***digitalWrite(10, HIGH);***

***digitalWrite(9, LOW);***

***digitalWrite(8, HIGH);***

***delay(retardo);***

***digitalWrite(11, HIGH);***

***digitalWrite(10, LOW);***

***digitalWrite(9, LOW);***

***digitalWrite(8, HIGH);***

***delay(retardo);***

***digitalWrite(11, HIGH);***

***digitalWrite(10, LOW);***

***digitalWrite(9, HIGH);***

***digitalWrite(8, LOW);***

***delay(retardo);***

***}***

***void apagado() { // Apagado del Motor***

***digitalWrite(11, LOW);***

***digitalWrite(10, LOW);***

***digitalWrite(9, LOW);***

***digitalWrite(8, LOW);***

}

Código en PSOC Creator:

***#include <ros.h>***

***#include <std\_msgs/Int16.h>***

***#include "StepperControl.h"***

***extern ros::NodeHandle nh;***

***namespace StepperControl {***

***const uint8\_t msec\_per\_step = 10;***

***int16\_t steps\_remaining = 0;***

***int16\_t abc = 0;***

***uint32\_t next\_time;***

***void messageCb( const std\_msgs::Int16& step\_msg){***

***steps\_remaining = step\_msg.data;***

***steps\_remaining = steps\_remaining\*1.4222222222;***

***}***

***ros::Subscriber<std\_msgs::Int16> sub("stepper\_move", &messageCb );***

***void setup()***

***{***

***nh.subscribe(sub);***

***next\_time = SysTimer::millis();***

***}***

***void paso\_der(){ // Pasos a la derecha***

***b2b\_Write(1);***

***b1b\_Write(0);***

***b2a\_Write(0);***

***b1a\_Write(1);***

***CyDelay(msec\_per\_step);***

***b2b\_Write(0);***

***b1b\_Write(1);***

***b2a\_Write(0);***

***b1a\_Write(1);***

***CyDelay(msec\_per\_step);***

***b2b\_Write(0);***

***b1b\_Write(1);***

***b2a\_Write(1);***

***b1a\_Write(0);***

***CyDelay(msec\_per\_step);***

***b2b\_Write(1);***

***b1b\_Write(0);***

***b2a\_Write(1);***

***b1a\_Write(0);***

***CyDelay(msec\_per\_step);***

***}***

***void paso\_izq() { // Pasos a la izquierda***

***b2b\_Write(0);***

***b1b\_Write(1);***

***b2a\_Write(1);***

***b1a\_Write(0);***

***CyDelay(msec\_per\_step);***

***b2b\_Write(0);***

***b1b\_Write(1);***

***b2a\_Write(0);***

***b1a\_Write(1);***

***CyDelay(msec\_per\_step);***

***b2b\_Write(1);***

***b1b\_Write(0);***

***b2a\_Write(0);***

***b1a\_Write(1);***

***CyDelay(msec\_per\_step);***

***b2b\_Write(1);***

***b1b\_Write(0);***

***b2a\_Write(1);***

***b1a\_Write(0);***

***CyDelay(msec\_per\_step);***

***}***

***void apagado() { // Apagado del Motor***

***b2b\_Write(0);***

***b1b\_Write(0);***

***b2a\_Write(0);***

***b1a\_Write(0);***

***}***

***void loop()***

***{***

***if ((int32\_t)(SysTimer::millis()-next\_time)>0) {***

***next\_time += msec\_per\_step;***

***if (steps\_remaining > abc) {***

***paso\_der();***

***abc++;***

***} else if (steps\_remaining < abc) {***

***paso\_izq();***

***abc--;***

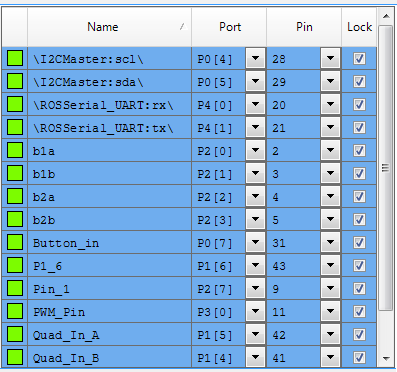
***}***

***}***

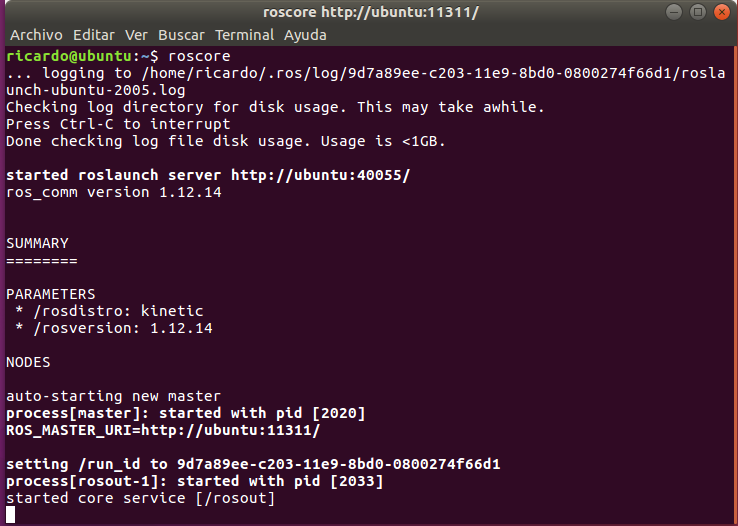
***}***

***} // namespace StepperControl***

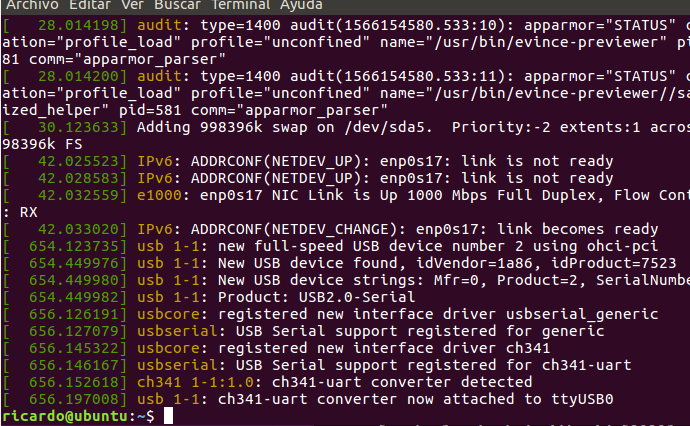
Mapeado de pines en PSoC



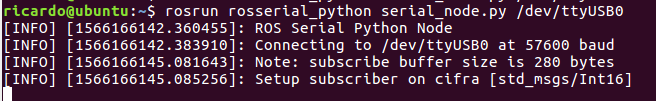
Inicialización de Ros en Ubuntu “roscore”



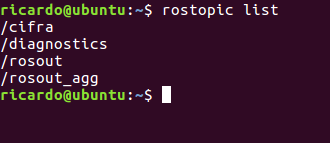
Comprobamos que el dispositivo se haya conectado correctamente con el comando “dmesg”



Iniciamos la comunicación con el comando “rosrun rosserial\_pythn serial\_node.py /dev/\*ubicación de nuestra tarjeta\*



Con el código rostopic list podemos ver que codigos tiene nuestra tarjeta



Como lo hemos nombrado “cifra” procederemos a ejecutarlo:



Después de int16 podemos poner los grados al que queremos llevar el motor, una vez terminado el proceso lo podremos terminar presionando “contrl+c” al mismo tiempo.

**Conclusión**

Con esta práctica logramos la comunicación entre ros y la tarjeta microcontroladora, lo cual nos permitirá controlar el brazo por coordenadas, o grados en este caso, al permitirnos manejar 3 nodos. Hay que cuidar que las bobinas se activen y desactiven de acuerdo al motor y driver que estamos usando, ya que al usar el primero que nos encontramos en internet no era el adecuado para nuestro motor, por lo que se debe ser muy cuidadoso para encontrar el que necesite para funcionar correctamente.