ROVER EXPLORADOR

Integrante 2 (Arial, 11 Pts, centrado)
e-mail: integrante2@institución (quitar hipervínculo)
Integrante 3 (Arial, 11 Pts, centrado)

RESUMEN: *En este documento se presenta el desarrollo de una plataforma robótica con control remoto usando el protocolo TCP/IP. Se describe las características físicas, mecánicas y eléctricas de la plataforma sobre la que se desarrolla ROVER Explorador y el modelo usado para su control con sus características. Se explica la decisión de la arquitectura de software empleada y como se implementó.*

PALABRAS CLAVE*: plataforma robótica, control remoto, Unix, C, streaming.*

1 INTRODUCCIÓN

El objetivo de ROVER Explorador es el diseño, construcción, implementación y depuración de una plataforma controlada remotamente y de streaming de video. Esto se consigue trabajando tanto en el área de la electrónica, como de la informática.

2 PLATAFORMA

Figura 1. Plataforma *Dagu Wild Thumber 6WD*.

Todo el desarrollo descrito en éste documento fue hecho sobre ésta plataforma robótica. A pesar de eso, gran parte de la implementación de software es independiente de la plataforma.

Los criterios usados, tales como: abrupto, terreno irregular y otros, son respecto a las dimensiones de la plataforma

2.1 CARACTERÍSTICAS FÍSICAS

La plataforma *Dagu Wild Thumber 6WD* cuenta con seis ruedas, cada una fija al eje de un motor DC (corriente continua). Cada motor es independiente del resto y tiene un grado de libertad, que permite que cada rueda pueda tener un ángulo de inclinación distinto al resto.

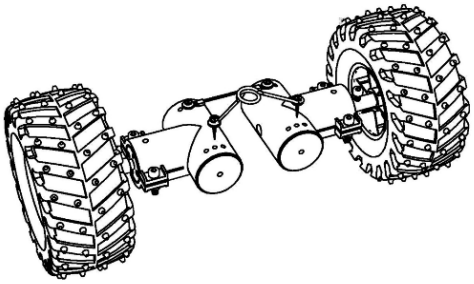

Figura 2. Diseño de par de motores *Wild Thumber 6V DC*.

Cada par frente a frente de motores está ligada con un cable metálico, que limita el ángulo máximo entre los dos motores, acotando por arriba la independencia entre cada par de motores. . Este diseño permite el desplazamiento por terreno irregular y superar obstáculos comparables con su altura en poco espacio

El chasis metálico en el que se fijan los motores es una placa con agujeros, los que se usan para fijar componentes electrónicos y otra placa plana que actúa de segundo nivel. Cuenta con dos contenedores donde se fijan baterías y más componentes electrónicos.

2.2 ELECTRÓNICA

La plataforma usa motores *Wild Thumber 6V DC* con una caja reductora 75:1. Éstos motores trabajan idealmente a 6[V], con rango de tolerancia de 2[V] a 7.5[V] como mínimo y máximo respectivamente, y un máximo de corriente de 5.5[A]. Soportan un torque de bloqueo de 8.8[Kg/cm], que es cuando consumen el máximo de corriente. Llegan a las 10.000[RPM].

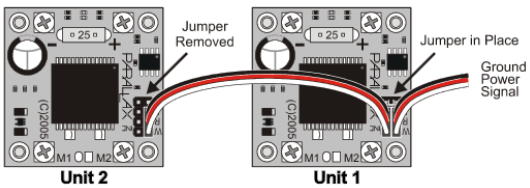

Figura 3. Conexión de modo dual de HB-25.

Los motores son alimentados por dos puentes H HB-25[7], cada uno controlando 3 motores. Éste modelo se caracteriza por poder ser controlados en serie, con lo que se tiene una cantidad de pines constante al aumentar el número de éstos. Soportan 25[A] y hasta

16[V] continuos, por lo que son soportan el control correcto de 3 de los motores.

Los HB-25 en conexión dual se controlan enviando una señal PWM al primero de la serie. Se envía un pulso entre 1[ms] y 2[ms]. Para máxima velocidad/fuerza de retroceso 1000[us], para estar quieto 1500[us] y 2000[us] para máximo avance. Entre cada señal hay un tiempo de *hold-off* de 5.25[ms], mientras que con un hold-off de 1.1[ms] se tienen los pulsos de cada puente H. El HB-25 arroja una señal pulso de ancho variable a los motores. El valor medio del voltaje es proporcional a la potencia entregada a los motores.

La señal de control es generada por una placa de desarrollo Arduino Mega. Se le carga un programa escrito en lenguaje C que implementa un protocolo de comunicación para el control de dos motores. Éste protocolo responde a las señales *Fwd(motorID*, *value) Rev(motorID, value)* y *Stop(motorID*. Debe cumplir con un valor de inicio y de fin de transmisión para detectar el paquete.

El Arduino es alimentado por una fit-PC2 a bordo, un computador portátil. La fit-PC2 tiene un procesador Atom de 1.6[GHz], 2 [Gb] de RAM y un disco duro SSD. Esto último es importante, ya que resiste movimientos bruscos, tales como golpes o caídas abruptas de la plataforma. Además, se le pude conectar un antena wifi. Es la fit-PC2 la que actúa como servidor, recibiendo instrucciones desde un cliente que envié mensajes con la estructura del estado del modelo explicado en 2.3.

2.3 MODELO TEÓRICO DE CONTROL

El modelo está basado en que para el control se tienen 2 motores virtuales, un motor izquierdo y un motor derecho, ambos formados por los tres motores del lado correspondiente.

El modelo diferencia dos velocidades. Una velocidad es la de avance, llamada *x* y la otra es la diferencia de velocidad entre el motor izquierdo y derecho, llamada velocidad *alfa*. Esto permite producir una tarea básica como un giro mientras se avanza de manera natural.

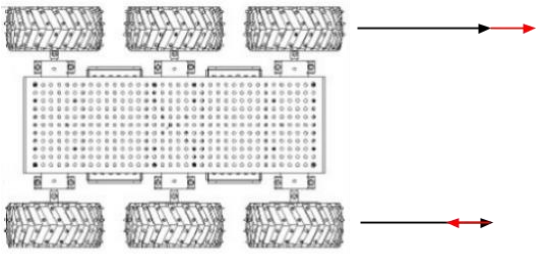

Figura 3. Representación de velocidades x (negra) y alfa (roja).

La figura 3 muestra una representación gráfica de ambas velocidades. El valor *alfa* está implementado, igualmente que el valor *x*, como un porcentaje, donde 100% representa la máxima diferencia de velocidad entre los motores y 0% la ausencia de diferencia, con lo que el robot se dirige hacia adelante. La implementación actual trunca en 100% y -100% en casos en que por la suma de la velocidad alfa, la velocidad resultante de los motores.

Éste modelo permite un control sencillo desde un joystick de juego común, ya que con sólo indicar un par *(x,y)* se puede generar ambas velocidad *x* y *alfa*.

3 SOFTWARE DE CONTROL

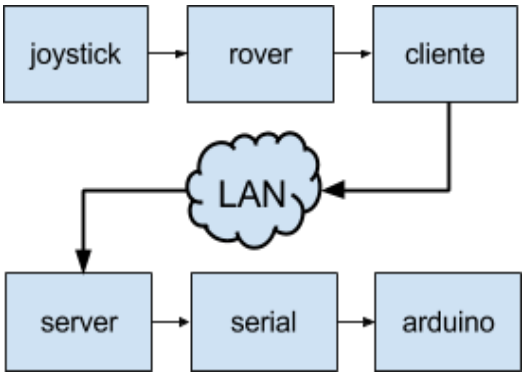

Figura 4. Diagrama de bloques de ROVER Explorador.

La figura 4 representa a alto nivel (alto grado de abstracción) la conexión realizada para el control de la plataforma. El cliente envía los comandos que luego se ejecutarán en la plataforma.

Ambas partes de la comunicación fueron desarrolladas usando el lenguaje C, sobre el sistema GNU/Linux, aprovechando tanto bibliotecas estándar del lenguaje, bibliotecas propias de GNU/Linux y comunicación de procesos, pipelines, estándar de los sistemas UNIX.

3.1 SEPARACIÓN EN PROCESOS

Se emplea la estrategia de producir un programa distinto por cada módulo presente en la figura 4. Una vez ejecutados, se convierten en procesos administrados por el sistema operativo.

El módulo “joystick” se preocupa de leer entradas “crudas” (raw) del joystick conectado. Se empleó la biblioteca proporcionada en Linux para su lectura[3]. En la figura 4 el módulo “joystick” representa tanto el programa encargado de usar la API de Linux, como a “js_adapter”. “js_adapter” toma los valores del módulo “joystick” y los adapta a “rover”. La existencia de éste módulo permite que el módulo “joystick” sea general y no entregue información útil sólo al módulo “rover”, sino también a cualquier futuro módulo a usar, con su respectivo adaptador.

“rover” es el módulo encargado de representar el modelo teórico de control descrito en el punto 2.3. Escribe los valores de velocidad *x* y *alfa* resultantes de la entrada leída de “js_adapter”.

El módulo “cliente” envía los valores a “servidor” mediante una conexión TCP/IP, ambas administrada por

el comando *netcat*. Esto es posible debido a como se comunican los módulos lo que se explica en el punto 3.2.

"serial" lee los valores del modelo de control y los convierte a comandos entendidos por el microcontrolador ATmega128 en una placa de desarrollo Arduino, descrito en 2.2. Se encarga, además, de su comunicación por el puerto serial(USB). COMUNICACIÓN ENTRE PROCESOS

UNIX entrega nativamente comunicación de procesos llamada *pipe*[4] (tuberías). En particular se usan *unnamed pipes* (tuberías anónimas). Esto se logra en una *shell*[6] como bash o zsh con la siguiente notación

programa1 | *program2* | *programa3* | …

Documentación al respecto se puede encontrar en [4] y [5].

Como exigencia se tiene que cada programa lea de la entrada estándar y escriba a la salida estándar. Esto permite entender el programa como un flujo de información siendo leída, procesada y escrita en cada programa.

De esta manera se consigue una alta independencia entre cada proceso, teniendo sólo que cumplir con el protocolo de comunicación, sin preocuparse por el estado interno de los demás procesos.

3.2 CLIENTE/SERVIDOR

Tanto el cliente (1) como el servidor (2) se implementan con el comando netcat, presente en todos los sistemas UNIX.

netcat –v rover.edu 54321 *(1)*

netcat –v –k –l –p 54321 *(2)*

netcat lee de la entrada estándar y escribe a la salida estandár, por lo que es posible usarlo, de modo natural, dentro de la cadena de *unnamed pipes* producidas con la *shell*.

3.3 STREAMING DE VIDEO

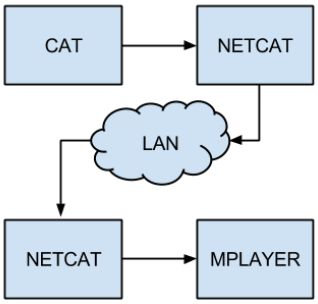

Figura 5. Diagrama de bloques de streaming de video.

La Webcam se conecta al fit-PC2. La información recolectada es escrita por el sistema operativo en el archivo */dev/video0* (Debian). Luego, el archivo *video0* es leído por el programa *cat,* que a su vez entrega éstos datos al programa *netcat*.

netcat es configurado para que convierta éstos datos en datagramas y los envíe por el protocolo IP/UDP al servidor. Por otro lado, se crea un *pipe*, donde *netcat* deja la información. También el programa *netcat* es puesto en modo *escucha* (modo listen) en un puerto alto elegido arbitrariamente y el reproductor de video *mplayer* es puesto en marcha para que reciba el streaming de video.

mkfifo a-mpjg; nc.traditional -lu -p 5000 > a.mjpg | mplayer -cache 32 -vo gl -demuxer lavf a.mjpg *(3)*

sudo cat /dev/video0 | nc.traditional -u [SERVER IP] 5000 *(4)*

En (3) se tiene la ejecución de del computador cliente, que recibe el streaming. El comando (4) se ejecuta al arrancar el servidor en la fit-PC2, produciendo un streaming de datos leídos por la Webcam.

3.4 IMPLEMENTACIÓN DE CONTROL

El resultado de lo descrito es que el control remoto se inicia al escribir lo siguiente en la *shell*

joystick|js_adapter|rover|netcat –v rover.edu 54321 (3)

netcat –v –k –l –p 54321 | serial (4)

En (3) se tiene la cadena de comandos para el cliente y en (4) la del servidor. Como se menciona en 3.1, "serial" se encarga de comunicarse con la placa Arduino.

4 REFERENCIAS

[1] Características DAGU WILD THUMPER http://www.dagurobot.com/manual/4WD%206WD%20short%20manual.pdf

[2] Características eléctricas y mecánicas "DAGU WILD THUMPER" http://www.dagurobot.com/goods.php?id=47

[3] Documentación de Linux joystick API https://www.kernel.org/doc/Documentation/input/joystick-api.txt

[4] Unix pipelines, Bell Labs http://cm.bell-labs.com/cm/cs/who/dmr/hist.html#pipes

[5] Explicación de Unix pipes http://web.cse.ohio-state.edu/~mamrak/CIS762/pipes_lab_notes.html

[6] FreeOS, resource center for free operating systems http://www.freeos.com/guides/lsst/ch01sec07.html

[7] HB-25 Datasheet http://www.parallax.com/sites/default/files/downloads/29144-HB-25-Motor-Controller-V1.2.pdf