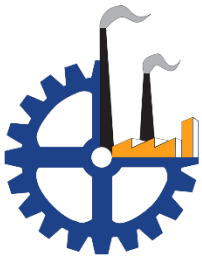

INSTITUTO TECNOLÓGICO DE CHIHUAHUA

REPORTE FEBRERO-MARZO
SERVICIO SOCIAL

Prototipo de robot omnidireccional con Raspberry Pi 4B controlado de forma remota.

Docente/Supervisor: Ing. Alfredo Chacón Almada

Participantes:

Marco Antonio Calderón Macías	20060915
Alejandro Morales Holguín	20060938

Contenido

Robot ArmPi Pro de Hiwonder.

El robot ArmPi Pro es un robot desarrollado por la empresa Hiwonder, su sistema principal consiste de una Raspberry Pi 4B, sus códigos de control están desarrollado en base a ROS y utiliza Python para programar sus distintas directivas, módulos y algoritmos. El robot cuenta con un brazo robótico de seis ejes junto a un sistema de visión por medio de una cámara USB, además cuenta con un chasis con ruedas mecánicas omnidireccionales controladas por medio de cuatro motores de corriente directa, gracias a esto el robot puede realizar funciones de movimiento omnidireccional, agarre por medio de su garra, seguimiento de objetos gracias a su sistema de visión, entre otras posibilidades.

El robot ArmPi Pro fue diseñado como herramienta didáctica de aprendizaje sobre robótica dirigida a los niños, por lo que el potencial de desarrollo del mismo se ve mermada debido a las limitaciones impuestas por la compañía a su software de control.

Raspberry 4B y placa de expansión.

Sistema operativo Ubuntu versión Hiwonder.

Cámara USB y visión computarizada por medio de OpenCV.

Chasis de ruedas omnidireccionales.

Brazo robótico de seis ejes.

ArmPi Pro PC Software.

Control remoto del robot por medio de protocolo HTTP.

Inicialización de un servidor HTTP mediante comandos '*bash*'.

Para inicializar un servidor HTTP en la Raspberry Pi se utilizó la herramienta el módulo de Python '*SimpleHTTPServer*', con el cual se pueden inicializar servidores en cualquier directorio del sistema. Para ejecutar el comando de inicialización del servidor, se tiene que ejecutar una terminal dentro de la carpeta destino donde se desee abrir el servidor o, en su defecto, utilizar el comando '*cd*' para redirigirse a dicho directorio. El comando de inicialización del servidor tiene la siguiente estructura:

```
python3 -m http.server PUERTO
```

Donde "*PUERTO*" corresponde de forma literal al número de puerto donde se desea alojar al servidor, en nuestro caso se eligió el puerto 8080 debido a su facilidad de escritura y a que el puerto 8000 ya se encuentra ocupado alojando los vídeos de las cámaras.

Para que el servidor HTTP se encuentre activo en todo momento este debe poder ejecutarse durante el arranque o '*boot*' del sistema Linux es necesario realizarlo mediante un script de '*bash*'. Bash es un '*shell*' de Unix, es decir, una interfaz de línea de comandos cuyo fin es el de interactuar con el sistema operativo. Este es el shell predeterminado en muchas distribuciones GNU/Linux. Su nombre es un acrónimo de Bourne Again SHell. Por convención se suele dar a este tipo de archivos la extensión '*.sh*'.

Un script de '*bash*' es un archivo de texto sin formato que contiene una serie de comandos. Estos comandos son una combinación entre los comandos que son normalmente utilizados en la terminal de Linux y otros comandos que podrían escribirse en la terminal pero que generalmente no suelen utilizarse. Cualquier comando que pueda ejecutarse normalmente en la terminal se puede poner en un script y realizará exactamente la misma acción, esto funciona en viceversa.

Para crear el script '*bash*' lo primero que se necesita es crear un archivo con la extensión '*.sh*' por medio de cualquier editor de texto que proporcione el sistema o se haya instalado en el mismo, en este caso

se optó por utilizar el editor *nano* debido a que ya se encontraba instalado en el sistema. El comando utilizado para la creación del script fue el siguiente:

```
nano NombreDelArchivo.sh
```

El script se puede componer de cuantas líneas sea necesario, pero resulta esencial la inclusión de la directiva del interprete o '*shebang line*', donde se indica el interprete o '*shell*' que utilizará el sistema para ejecutar los comandos. La directiva del interprete comienza con un símbolo de etiquete o coloquialmente conocido como gato (#), seguido de un signo de exclamación (!). Posterior al signo de exclamación, se debe especificar la ruta al intérprete que se desea usar. Ejemplo:

```
#!/bin/bash
```

Como ya se mencionó, para ejecutar el servidor HTTP resulta necesario ubicarse en la carpeta destino, es por eso que el primer comando del script es un cambio de directorio hasta la carpeta destino, posterior a este comando se colocó el comando de inicialización del servidor.

```
cd /home/ubuntu/www/
python3 -m http.server 8080
```

Luego de crear y guardar el contenido del script '*bash*' se necesita otorgarle premisos de ejecución para todos los usuarios del sistema; para esto se hizo uso del comando '*chmod*', utilizando el parámetro '*x*' que corresponde a los permisos de ejecución.

```
chmod +x NombreDelArchivo.sh
```

Sin embargo, para lograr que este script se ejecute durante el arranque del sistema se tiene que utilizar otra herramienta de Linux llamada '*systemd*' Systemd es un conjunto de componentes básicos para un sistema Linux. Proporciona un administrador de sistemas y servicios que se ejecuta e inicia el resto del sistema, es gracias a esta herramienta que se pueden inicializar varios drivers para los periféricos de la

Raspberry Pi y también es gracias a esto que se inicializan muchas funcionalidades del sistema de Hiwonder.

Systemd incluye utilidades para controlar la configuración básica del sistema como el nombre de host, la fecha, la configuración regional, mantener una lista de usuarios conectados y contenedores y máquinas virtuales en ejecución, cuentas del sistema, directorios y configuraciones de tiempo de ejecución, y servicios para administrar una red simple. configuración, sincronización horaria de la red, reenvío de registros y resolución de nombres.

Relacionado a esto último, se busca convertir al script del servidor HTTP en un servicio más del sistema, para lo cual se comienza creando un archivo con extensión '*.service*' en el directorio donde se alojan todos los servicios: /etc/systemd/system/. Para esta acción se requieren permisos de super usuario, por lo que se utilizó el siguiente comando:

```
sudo nano /etc/systemd/system/NombreDelArchivo.service
```

Dentro de este archivo se escriben los siguientes parámetros:

```
[Unit]
Description=Starts a HTTP server on port 8080
After=network.target

[Service]
ExecStart=/home/ubuntu/www/NombreDelArchivo.sh

[Install]
WantedBy=default.target
```

En el apartado titulado como '*Unit*' se proporciona metadatos descriptivos sobre el servicio en cuestión.

- '*Description*': Es una descripción legible para los usuarios. En este caso, describe la acción literal del servicio que es iniciar un servidor HTTP en el puerto 8080.
- '*After*': Especifica las dependencias que deben iniciarse antes de este servicio. En este caso, indica que el servicio debe iniciarse posterior al servicio de red (*network.target*).

En el apartado titulado como '*Service*' se define cómo debe ejecutarse el servicio.

- '*ExecStart*': Especifica la ruta del script para iniciar el servicio. En este caso, se proporciona la ruta al script '*bash*' que se desea ejecutar.

Por último, en el apartado titulado como '*Install*' se define cómo se instalará y habilitará el servicio en el sistema.

- '*WantedBy*': Indica el objetivo que quiere este servicio. '*default.target*' es el objetivo predeterminado del sistema.

Como paso final, para que el servidor HTTP pueda ser ejecutado en el arranque del sistema, se requiere cargar el nuevo archivo de servicio en el sistema, habilitarlo e iniciarlo. Para esto se utilizan los siguientes comandos en una terminal:

```
sudo systemctl daemon-reload
sudo systemctl enable myscript.service
sudo systemctl start myscript.service
```