TFG - Robot Android y Telepresencia.

***Desarrollo de un robot de telepresencia con Android*.**

Android Robot and telepresence

Author: Dailos Reyes Diaz
Departamento de Ingeniería de Sistemas y Automática y Arquitectura de Computadores.
Escuela Técnica Superior de Ingeniería Informática
Trabajo de Fin de Grado

La Laguna, 07 de julio de 2014

D. **Jesús Miguel Torres Jorge**, con N.I.F. 43826207-Y, Doctor adscrito al Departamento de Ingeniería de Sistema de Automática y Arquitectura de Computadores de la Universidad de La Laguna

D. **José Demetrio Piñeiro Vera**, con N.I.F. 43774048-B, Doctor adscrito al Departamento de Ingeniería de Sistema de Automática y Arquitectura de Computadores de la Universidad de La Laguna

C E R T I F I C A

Que la presente memoria titulada:

"Robot Android y Telepresencia."

ha sido realizada bajo su dirección por D. Dailos Reyes Diaz, con N.I.F. 54053463-J.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de Julio de 2014

Agradecimientos

Don Jesús Miguel Torres Jorge por su implicación con el proyecto y su dirección.

Don Demetrio Piñeiro Vera por su codirección y su apoyo al proyecto.

La universidad de la laguna por brindarme la oportunidad de desarrollar este proyecto.

Mis familiares y amigos por el apoyo recibido durante el desarrollo del mismo.

Resumen

El objetivo de este trabajo de fin de grado ha sido el desarrollo de un robot móvil con telepresencia usando como CPU un móvil con sistema operativo Android.

Para ello se toma como base un robot muy simple comercial, de la marca Pololu y se amplía añadiéndole sensores y una plataforma para la colocación de un dispositivo android que hará de "cerebro" ofreciendo al robot sus características y periféricos tales como la cámara, el gps, o los demás sensores propios del teléfono.

Palabras clave

Robot, Android, Telepresencia, Sensores, Automática.

Abstract

This work of degree´s end, its mean to be a mobile robot with telepresence. In addition, the robot use an android device installed in a platform that will serve as the "brain" of the robot. Doing this, we can make the robot more "intelligent" adding it web capabilities and the use of all the device sensors, such as the camera or the gps.

Keywords

Telepresence, Robot, Android, Sensors

Índice general

Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1

1. INTRODUCCIÓN

1.1. ANTECEDENTES

Robot "Curiosity" de la NASA.

Existen multitud de ejemplos de robots con telepresencia utilizados a diario en muchos campos. Uno de ellos quizá el más famoso es el robot "Curiosity" de la NASA, el cual permite tomar imágenes de su actividad en Marte e interactuar con su entorno de manera remota.

Como aproximación a nuestro proyecto existe un robot llamado "Botiful", que básicamente utiliza el dispositivo para comunicarse con el usuario a través de Skype, permitiendo al robot moverse con el usuario a la vez que se mantiene una conversación.

http://www.botiful.me/

1.2. OBJETIVOS

Lo primero que cualquiera puede pensar a la hora de echar un vistazo a este proyecto es, ¿para qué sirve? ¿qué función tiene?
En principio el trabajo consiste un robot móvil con las tareas que se le piden a cualquier dispositivo con telepresencia. Tales tareas podrían ser la de ofrecer visión y movilidad en terrenos hostiles o de difícil acceso a los seres humanos.

El principal objetivo para el proyecto consiste en conseguir un robot funcional que utilice un dispositivo Android como cerebro, dando órdenes a los actuadores del robot según las reciba desde un lugar remoto.

A parte del objetivo principal, se pretende añadir al robot todos aquellos sensores que le faciliten una cierta autonomía y le permitan mejorar la navegación y posicionamiento.

1.3. EXPECTATIVAS

Las expectativas con este proyecto son las de mejorar conocimientos personales acerca de la robótica en general, es un proyecto del que se espera poder poner en práctica todos estos conocimientos, y capaz de aportar nuevas experiencias y conocimientos acerca del tema.

Capítulo 2

2. LA BASE PARA EL ROBOT: POLOLU

Logo de "Pololu"

2.1. LA COMPAÑÍA POLOLU

Pololu es un fabricante y tienda online de componentes electrónicos, que vende a todo tipo de clientes, especialmente artículos relacionados con la robótica, como pueden ser sensores, motores, o robots.

Robot "3pi", de Pololu

2.2. EL 3PI

Dentro de la gama de robots, Pololu distribuye el "3pi", un robot móvil con dos ruedas, montado sobre el chip "Atmega 328", similar al de un ARDUINO UNO, el cual es descrito en profundidad más adelante, en el capítulo de materiales y recursos. Dicho chip permite desarrollar el código para el robot del proyecto utilizando el IDE de ARDUINO.

El 3pi es un robot que se suele usar en campeonatos de robots sigue líneas gracias a los 5 sensores de reflectancia que lleva en su parte inferior. Con ellos es capaz de distinguir de manera muy precisa caminos oscuros sobre fondos claros.

Se ha tomado la decisión de usar el 3pi para simplificar el trabajo. Únicamente serán usados del robot original sus motores, sistema de alimentación y chasis. A modo de sugerencia, como modelo más personalizado de este TFG se podría sustituir el 3pi por un chip de la familia ATMEL sobre un chasis fabricado desde cero.

CAPÍTULO 3

3. MATERIALES Y RECURSOS

3.1. EL M3PI

Variante "M3pi" del robot de Pololu

En el capitulo anterior se habla acerca del robot utilizado como base para el proyecto, el "3pi". Esta versión llamada "M3pi" es la que ha sido proporcionada por la universidad y se trata de un "3pi" con un módulo de expansión que encaja en la parte superior y da la capacidad de añadir al mismo un microcontrolador MBED el cual añadiría potencia extra a los programas para el "3pi" ya que contiene un procesador ARM capaz de procesar programas muchos más complejos y almacenar una gran cantidad de datos en comparación al ATMEL que viene con el 3pi básico.

En principio esto suena muy bien pero para el proyecto no es necesario este microcontrolador, pues se dispone de un dispositivo Android que ya dispone de un procesador ARM, el cual dependiendo de la gama del dispositivo va a dar una potencia de cómputo al robot muy superior a la que es posible obtener con el M3pi.

Dicho esto, aclarar el uso que se le da al modulo de expansión y al MBED:

3.1.1. MBED COMO PROGRAMADOR:

El modulo de expansión y el MBED se han considerado útiles a la hora de resolver uno de los primeros problemas encontrados; la carga del "bootloader" de "Arduino" en el ATMEL del "3pi".

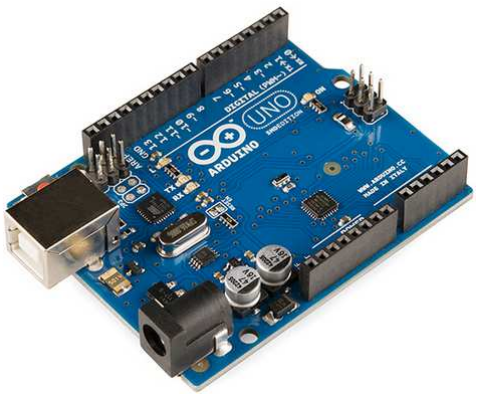
Este problema surge debido a que los programas compilados por el IDE de "Arduino" (el cual se utiliza en este proyecto) necesitan de uno de los dos métodos posibles para ser cargados en el "3pi".

El primero de estos métodos es el de cargar un "bootloader" de "Arduino" que ocuparía siempre una parte de la memoria del ATMEL. Este "booloader" se encargaría de reconocer los programas que se cargan en la parte restante de la memoria como programas compatibles con "Arduino" y los ejecutaría.

El segundo método trata de usar un programador para cargar los "sketch" de "Arduino" directamente al "3pi" por el puerto ISP. Este método se considera mas rápido que el anterior pues el "3pi" no necesita cargar el "bootloader" cada vez que arranca y se ahorra ese espacio que ocupa en la escasa memoria que ofrece el ATMEGA 328.

A pesar de las ventajas del segundo método se ha considerado más cómodo el uso del primer método al no disponer de un programador AVR, contando en su lugar, con un MBED que haría de programador para cargar el "bootloader" y cargar luego simplemente los "sketchs" como si de un "Arduino" se tratase por serial. Más adelante en el capítulo del desarrollo se comenta porque ha fallado este método y se ha tenido que recurrir al segundo.

3.2. ARDUINO



Placa "Arduino UNO"

"Arduino" es una conocida herramienta hardware de código libre que no es más que un ATMEGA 328 similar al del "3pi", pero integrado en una placa que ofrece tanto salidas y entradas digitales como entradas analógicas.

"Arduino" es una herramienta que permite llevar a cabo todo tipo de proyectos de robótica, demótica, y electrónica en general.

Una de las grandes ventajas de "Arduino" es la simplicidad software. Pues cuenta con un IDE también open source que permite la compilación, carga, edición y depuración de todos los programas, escritos en "wiring". Este IDE además tiene incluidas muchas librerías que facilitan el trabajo de mover Servos y otros motores, así como leer de sensores analógicos y digitales.

Quizá la mayor ventaja de esta herramienta con respecto a otras alternativas reside en su amplia comunidad. Muchísima gente utiliza "Arduino" y es fácil resolver cualquier duda si se acude a la comunidad.

3.2.1. "ARDUINO" COMO PLACA DE DESARROLLO

En este proyecto, se utiliza "Arduino", tanto su IDE como la placa física en sí.

Por un lado todo el código que se genera para el 3pi será compilado en dicho IDE y cargado al 3pi mediante el programador AVR de que se dispone.

Por otro lado, "Arduino" sirve para modelar los sensores que van a ser incluidos en el robot principal. Por ejemplo, para el calibrado del sensor sharp (distancia de objetos) se utiliza un robot en miniatura conectado directamente a "Arduino", codificando todas sus funciones. Este código, una vez optimizado es totalmente compatible con el programa principal que usa el 3pi.

3.3. PROGRAMADOR USBASP

Dispositivo USBAsp de la marca "BAITE"

Este dispositivo permite escribir en la memoria de otros dispositivos que no dispongan de una interfaz USB directa (como es el caso del "3pi"). Lo que hace es sustituir la memoria del ATMEGA 328 con un programa compilado, sobrescribiendo todos los datos.

Este modelo en concreto es de 10 pines, sin embargo el puerto ISP del 3pi es de tan solo 6 puertos. Más adelante se describe como realizar la adaptación.

3.3.1. USBASP COMO CARGADOR DE PROGRAMAS

Este será el método de carga de los "sketchs" en el 3pi seleccionado. ya que no pudo realizarse el plan original de la carga del "bootloader".
Al conectar el "USBAsp" al "3pi" se debe encender, pues el "USBAsp" utiliza la alimentación del mismo.

3.4. DISPOSITIVO ANDROID

Teléfonos Móviles con Sistema Operativo "Android"

Hoy en día, casi es muy común ser propietario de un teléfono móvil de los denominados "Smartphone", la mayor parte del mercado de estos dispositivos corresponde a los que utilizan el sistema operativo Android, El cual fue desarrollado por la empresa Android Inc. la cual fue financiada por GOOGLE y comprada por ésta mas tarde.

La mayor parte del código de Android ha sido liberada por Google bajo licencia apache.

La mayoría de los dispositivos móviles que funcionan con este sistema operativo, y cada vez en mayor medida, son auténticos computadores con una potencia de cálculo notable, teniendo la mayoría de estos procesadores de varios núcleos y una gran memoria principal.

Estos dispositivos cuentan con un gran número de sensores interesantes desde el punto de vista de la robótica, como son el giroscopio, el acelerómetro, la cámara, y el GPS.

3.4.1. TELÉFONO MÓVIL COMO CEREBRO.

El dispositivo Android va a ser utilizado por el sistema de este proyecto como centro de operaciones para el "3pi", manteniendo una comunicación serial con éste.

El móvil decide como actúa el 3pi, y envía órdenes a sus actuadores a través del puerto serial. También es capaz de leer la información que le transmiten los sensores con este mismo método.

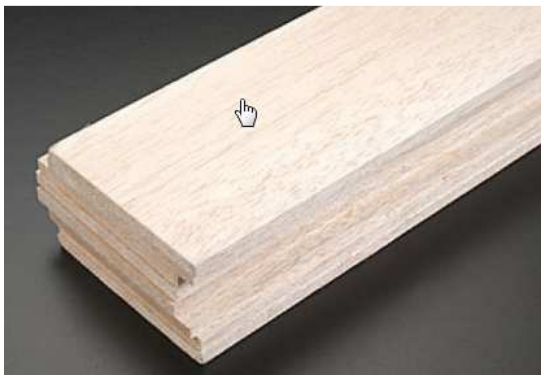
En otro capítulo se describe el desarrollo del protocolo utilizado para la comunicación.

3.5. CP2102 PUENTE USB-SERIAL

Puente USB-Serial de la marca "BAITE"

Este dispositivo USB se encarga de hacer compatible el intercambio de información entre un dispositivo que solo dispone de dos líneas seriales "rx", y "tx" como es el caso del "3pi", y un computador más complejo con un host USB como es nuestro teléfono móvil "Android".

3.6. MADERA DE BALSA

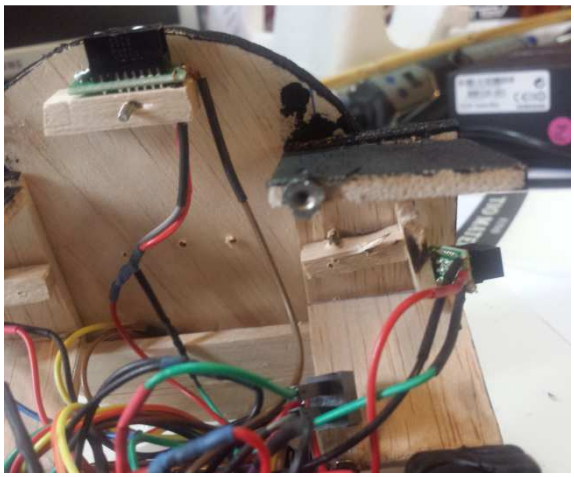


Láminas de madera de balsa.

La madera de balsa es un material extremadamente ligero utilizado frecuentemente en aeromodelismo, un campo en el propiedades como la ligereza son esenciales.

Se trata de una madera fácilmente manejable que permite hacer todo tipo de estructuras solidas muy apropiadas para este tipo de robots de tamaño reducido.

3.6.1. CHASIS A BASE DE MADERA DE BALSA



Vista inferior del chasis del "3pi"

Se ha utilizado este material para acomodar todos los sensores que se han añadido a al proyecto, ocultar el cableado y diseñar la plataforma para el teléfono "Android".

El resultado ha sido muy satisfactorio debido a que se obtiene una estructura solida y elegante sobre la que trabajar sin problemas.

3.7. SENSORES SHARP GP2Y0D805Z0F DIGITAL 5CM

Sensor Sharp de 5cm

Se trata de un sensor de proximidad digital, se activa cuando encuentra algún obstáculo en su rango de visión, que son 5 centímetros en línea recta.

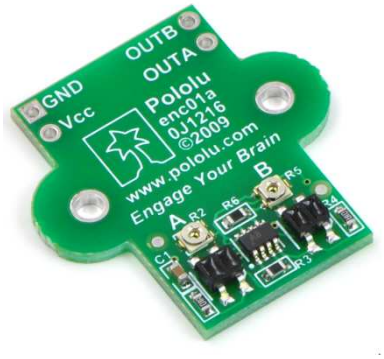
Es un sensor con una respuesta muy rápida, ideal para usar como seguridad anti choques.

3.7.1. SISTEMA ANTI CHOQUES, SENSORES SHARP.

El robot contará con tres de estos sensores, uno al frente de su chasis circular, otro a la izquierda, y otro a la derecha.

De este modo, el robot será capaz de evitar obstáculos que hagan que se quede bloqueado en algún lugar. Un sistema como éste es básico si se pretende dotar al sistema de cierta autonomía.

3.8. POLOLU WHEEL ENCODERS.



Encoders para las ruedas

Estos encoders que proporciona Pololu, constan de dos sensores de reflectancia que al ser puestos a escasa distancia del interior de una rueda dentada, serán capaces de medir cada uno de los dientes de la rueda.

3.8.1. SISTEMA DE CONTROL DE VELOCIDAD Y DISTANCIA

Gracias a los encoders, es posible llevar un control preciso de la velocidad que se desea adoptar por el robot, así como de la distancia recorrida.

Los encoders miden información de velocidad en las ruedas y van regulando la potencia de los motores hasta que la velocidad medida corresponda con la deseada.

3.9. RUEDA DE 42X19MM

Rueda con interior dentado para los encoders

Se trata de una rueda maciza con el interior dentado, ideal para encajar el encoder y obtener buenas medidas.

3.10. SENSOR SHARP GP2Y0A02YK0F ANALÓGICO, DE 20 A 150 CM

Sensor Sharp de 150cm

Este tipo de sensores, son similares a los anteriores Sharp de 5cm pero no se limitan a ofrecer una señal digital a la hora de detectar un obstáculo en su rango de visión, sino que se trata de un sensor analógico, el cual aporta información a cerca de a qué distancia se encuentra dicho obstáculo.

Su rango de trabajo es desde los 14 centímetros aproximadamente, hasta el metro y medio.

Este sensor se encuentra en clara desventaja con respecto a un sistema de telemetría convencional en robots móviles más grandes, como puede ser un telemetro laser, debido a que solo es capaz de medir en línea recta. Sin embargo se ha diseñado un método para reforzar esta desventaja e imitar el funcionamiento de dichos telémetros.

3.10.1. SISTEMA DE MAPEADO.

Con este sensor Sharp se pretende dotar al robot de un sistema de mapeado, capaz de reconocer su entorno. Más adelante se detalla cómo ha sido posible llevar esta idea a la práctica.

3.11. SERVO TOWERPRO SG92R

Servo Motor, tipo "micro"

Los servo motores son los mejores aliados en cualquier proyecto de robótica basada en manipuladores.

Se trata de un motor de continua, con una reductora, una controladora y un potenciómetro.

La principal ventaja de estos motores es que se le puede especificar su posición, que en la mayoría de los casos, estará en un rango de 180º grados. En el caso de este modelo, el rango está limitado a los 170º grados, con lo cual hay que tomar en consideración ciertas operaciones para su correcto funcionamiento, y además, limitar el rango de detección.

Al tratarse de un servo analógico, hay que proporcionarle la posición en forma de señal PWM la cual según su frecuencia hará que el motor esté en una posición u otra.

3.11.1. CABEZAL DE GIRO PARA EL MAPEADO

Si se añade el giro de este servo a la detección en línea recta del sensor Sharp, es posible hacer barridos en esta amplitud para obtener información de su entorno tal y como lo haría un telemetro laser, en sus 180º de rango.

Así pues, se monta el sensor Sharp de 150cm encima del eje de este servo para lograr un mapeado más útil para el robot.

3.12. RECURSOS SOFTWARE

3.12.1. TEAMVIEWER

Teamviewer es una aplicación gratuita de control remoto, que en algunos dispositivos Android sirve para controlar el móvil de manera remota, evitando en principio (con las desventajas de fluidez que ello conlleva) utilizar un software remoto de control.

3.12.2. "WIRING" (ARDUINO)

Todo el código que lleva el 3pi en su interior ha sido desarrollado en "Wiring". tanto los controles para los actuadores como el protocolo de comunicación serial del 3pi con el dispositivo Android.

"Wiring" es un lenguaje de programación diseñado para encapsular todas las funciones, variables y funcionalidades propias del hardware. Este lenguaje, además de facilitar este trabajo, permite usar las características de C++.

3.12.3. JAVA

Se utiliza Java para la elaboración de un software de conexión remota, mediante sockets encargado de enviar ordenes al dispositivo Android.

3.12.4. ANDROID APP

Se ha desarrollado una Aplicación para Android, encargada de enviar y recibir información directamente con el 3pi, utilizando una librería en Java para la comunicación serial.

Capítulo 4

4. EL 3PI, Y EL ENTORNO ARDUINO

4.1. CONOCIENDO AL 3PI

"3pi" sigue líneas

La primera fase previa al desarrollo del proyecto ha sido la de la familiarización con el 3pi. Para ello practica escribiendo varios códigos para MBED, utilizando el editor online que se facilita en la URL http://mbed.org/

Por fortuna, la interfaz de MBED resulta muy fácil de utilizar ya que se codifica directamente en el editor online usando C/C++ y se genera un ejecutable binario ya compilado, el cual basta con arrastrar al dispositivo USB que aparece al conectar el MBED al PC.

Uno de los ejemplos encontrados para practicar con el "m3pi" es un sigue líneas implementado con un PID, diseñado para mantener al "3pi" siguiendo las líneas a la vez que acelera hasta su velocidad punta siempre que tiene oportunidad.

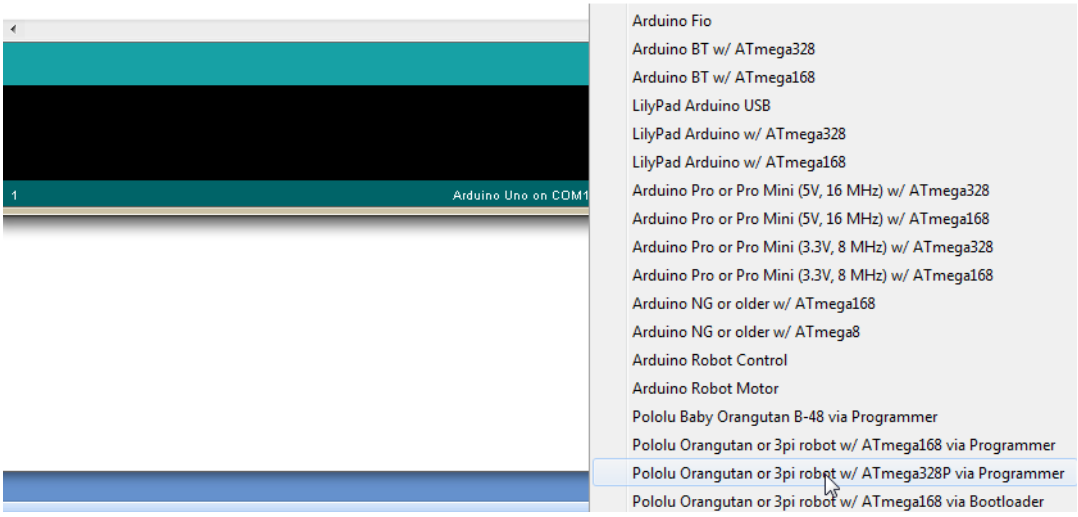
Como ya se ha explicado anteriormente, el "3pi" está diseñado para ser utilizado en este tipo de campeonatos donde los participantes escriben sus propias implementaciones del sigue líneas (*véase la figura superior)*, haciendo uso de los 5 sensores de reflectancia que posee el "3pi" en su parte inferior. Estos sensores son similares a los dos que presenta el encoder utilizado en el proyecto, y son capaces de distinguir superficies oscuras de otras más claras.

4.2. PREPARANDO LA COMPATIBILIDAD CON EL ENTORNO DE ARDUINO

4.2.1. PREPARACIÓN DEL ENTORNO ARDUINO

Lo primero que se debe hacer es preparar el IDE de ARDUINO para que sea compatible con los productos de Pololu, para ello se encuentra disponible un parche en la web de Pololu, el cual debemos instalar en el directorio de ARDUINO.

Una vez seguidos los pasos, el IDE debería mostrar la opción de cargar "sketchs" directamente a un robot Pololu.



IDE de Arduino con el parche para Pololu

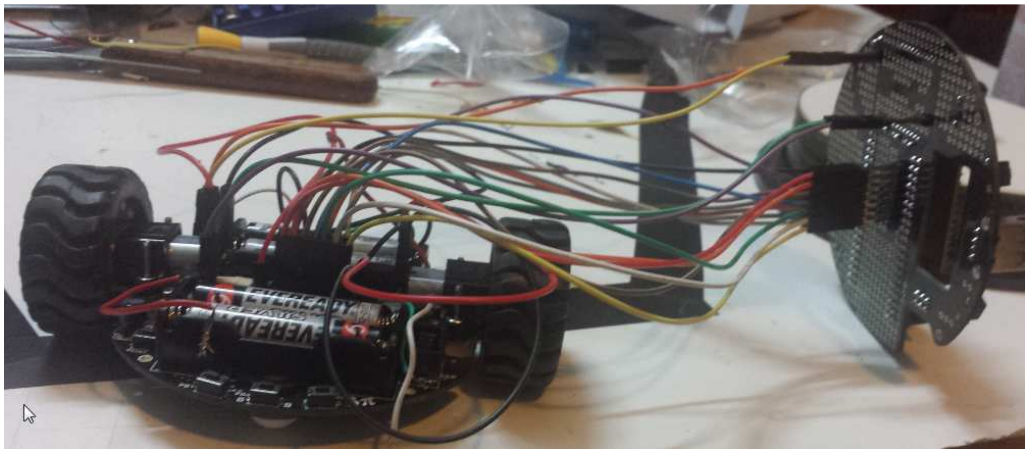
4.2.2. PRIMERA ALTERNATIVA: CARGAR EL "BOOTLOADER"

Se dispone de un "3pi" y el "MBED" para comenzar a trabajar, y se desea poder cargar "sketchs" de ARDUINO desde su propio IDE, véase la referencia a la función del "MBED" en el capítulo de materiales y recursos.

Las placas ARDUINO utilizan lo que se conoce como "bootloader" para cargar los programas, que viene a ser una porción de código cargado en la primera zona de la memoria, el cual dispone de la información necesaria para hacer funcionar los programas que se le indique en el resto de la memoria. Para cargar este "bootloader" buscamos un programa en "MBED" que le permita actuar como programador ISP.

El programa de "MBED" utilizado espera un fichero binario en la raíz del dispositivo para cargarlo como "bootloader" al 3pi. Si el resultado funciona, el programa de ejemplo que viene con el "3pi" será sustituido por el "bootloader" de ARDUINO, y se podrá empezar a cargar nuestros "sketchs".

Este "bootloader" se encuentra en el directorio de instalación de ARDUINO, pero aparece compilado en hexadecimal. Como el "MBED" solo acepta ficheros binarios, se debe hacer uso de algún traductor que pase el fichero hex a BIN. Con éste preparado, conectamos el MBED para que se comunique con el ATMEGA del "3pi" mediante el puerto ISP.



3pi y MBED conectados

Una vez preparado el entorno, al intentar cargar el "bootloader" aparece el primero de los problemas. El "MBED" devolvía por serial que no reconocía el chip del "3pi".

Después de analizar bien todo el código se extrae que el pin de RESET en el 3pi es distinto al pin que asigna por defecto el código del programador en el "MBED", se corrige este detalle, se recompila y comienza a funcionar.

```
Binary file opened successfully
mbed AVR910 Programmer starting
el response de la 143: 53Enable programming command successful
Microcontroller is an Atmel [0x1e]Part family and flash size code is: 0x95Part number code is: 0x0f
Page 0 written
Page 1 written
.
.
.
.
.
Page 253 written
Page 254 written
Programming was successful!
```

"MBED" cargando el "Bootloader"

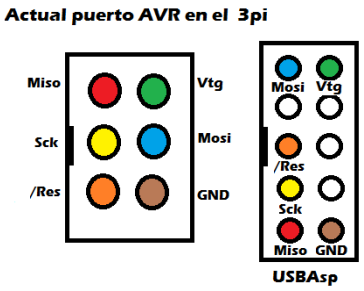
Una vez cargado, ya debería de ser posible la carga de los "sketch" por serial directamente desde un puente USB-Serial, véase referencia a el puente USB-Serial en la sección de materiales y recursos, o puenteando el "tx" "rx" de una placa ARDUINO hacia el 3pi.

Al intentar dicha carga, aparece un nuevo problema. Esta vez el AVRdude (programa que se encarga de la escritura hacia un dispositivo AVR) da un error de sincronización. Error que se debe a que en un ARDUINO, para que un "bootloader" cargue un "sketch", la línea del reset es puesta a baja directamente mediante una interconexión con la línea DTR, mientras que en el 3pi no hay nada conectado al reset.

Este problema lleva a desechar definitivamente la idea de usar un "bootloader", pues la solución requeriría la modificación de ciertas líneas en el circuito impreso del "3pi". Dicha solución se considera inviable en el proyecto y ante la nueva perspectiva se toma como opción definitiva el uso de un programador.

4.2.3. OPCIÓN DEFINITIVA, USO DEL USBASP

Después de desechar la idea de usar el "bootloader" de ARDUINO, se pasa a la colocación del programador AVR del que se dispone, véase referencia al "USBAsp" en la sección de materiales y recursos. Dispositivo en el que como se ha mencionado existen 10 pines, haciéndolo directamente incompatible con el conector ISP de 6 pines que se encuentra en el 3pi. Para solucionarlo, se adaptan las interfaces cruzando algunos pines de ambos conectores y dejando libres aquellos pines del "USBAsp" que no son necesarios, este trabajo se ilustra en la figura que viene a continuación.



Esquema de adaptación ISP

Una vez conectado, basta con seleccionar en el IDE de Arduino, el "3pi" mediante programador, y seleccionar el "USBAsp" como modelo de programador, dejando preparado al sistema para la carga de cualquier "sketch".

Capítulo 5

5. EL CHASIS Y LAS NUEVAS RUEDAS CON LOS ENCODER.

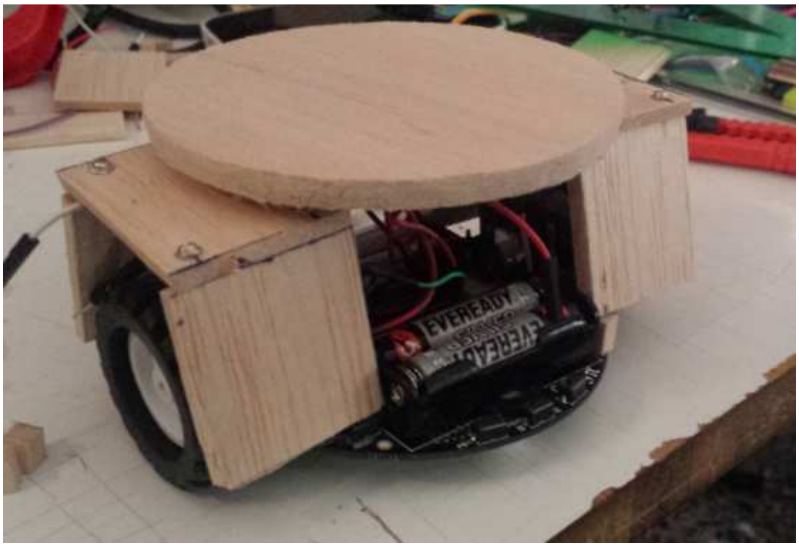
5.1. CONSTRUCCIÓN DEL CHASIS

Lo primero que se ha tenido en cuenta a la hora de diseñar un chasis para el robot, es orientarlo a la posterior creación de una base en su parte superior donde está ubicado el dispositivo Android, (*véase la referencia a los dispositivos Android en la sección de materiales y recursos).*

Con esto en mente, se comienza a cortar la base circular de la estructura como se muestra en la figura inferior, adoptando un diámetro ligeramente superior al de la propia placa del "3pi", dicha holgura en el diámetro de la base, permitirá una mejor colocación de todos los sensores, ya que se necesita de cierto espacio para ellos tal y como se muestra más adelante.

Corte circular de la base del chasis

A continuación se recortan los "guardabarros" que serán los encargados de por un lado albergar los sensores Sharp laterales, así como de aguantar el resto del chasis sujeto a la placa del "3pi", quedando el prototipo de la estructura como muestra la figura inferior, donde se ilustra la idea de ocultar el cableado, y se puede apreciar como la base es ligeramente más ancha que la placa del 3pi. Obsérvese también en la figura, como la estructura encaja gracias a la colocación de orificios en la madera para albergar las cabezas de los tornillos que hacen relieve en las superficies de la estructura, mostrando asi lo moldeable que este material, y lo útil que resulta para la elaboración de este tipo de estructuras.



Prototipo de chasis

Los próximos elementos del chasis serán añadidos según sean requeridos por los diferentes elementos faltantes.

5.2. COLOCACIÓN DE LAS NUEVAS RUEDAS

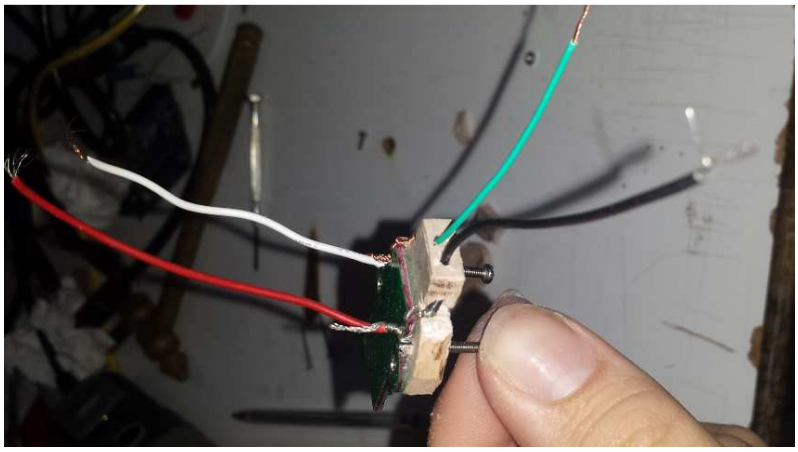
Las ruedas con el interior dentado que vamos a utilizar en el 3pi son notablemente más grandes que las originales, y no están preparadas para ser simplemente sustituidas, puesto que desequilibrarían al robot al tener mayor diámetro, obligándonos a diseñar unas plataformas extra sobre las que irán montadas las placas de los encoder, y sobre ellas los motores con sus soportes. Estas plataformas han sido elaboradas con madera de balsa, al igual que la estructura principal.

El primer paso requerido antes de el montaje de las plataformas consiste en limar una parte de las placas de los encoder, ya que necesitamos que su parte curva, que en dicha placa

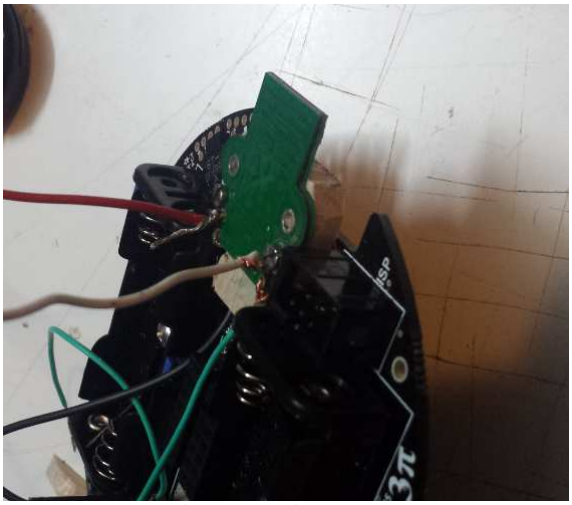
coincide con la esquina donde ésta se ensancha, sea más estrecha para que los tornillos que van a unirla con la placa del "3pi" encajen con ella.

A continuación, tal y como se ve en la primera de las figuras inferiores, se une la plataforma con el encoder insertando el cableado por las vias perforadas en ésta, dejandola lista para ser conectada, y unida a la placa del "3pi" tal y como se muestra en la segunda figura inferior.

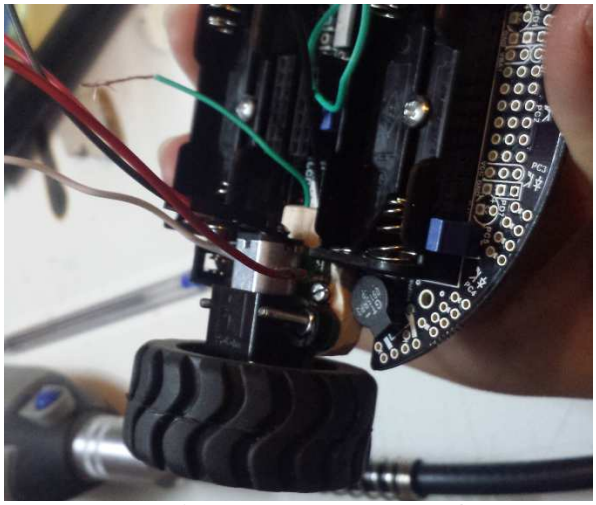
Una vez unidas la plataforma con la placa del "3pi" se puede observar en la tercera figura inferior como el grosor de la plataforma ha sido cuidadosamente seleccionado para que la nueva rueda quede al nivel de la bola de equilibrado que posee el "3pi" en su parte inferior, y que de no existir ésta plataforma, el robot quedaría inclinado hacia dicha bola.



Primera figura; Encoder y plataforma



Segunda figura; Plataforma en el "3pi"



Tercera figura; Resultado de la colocación

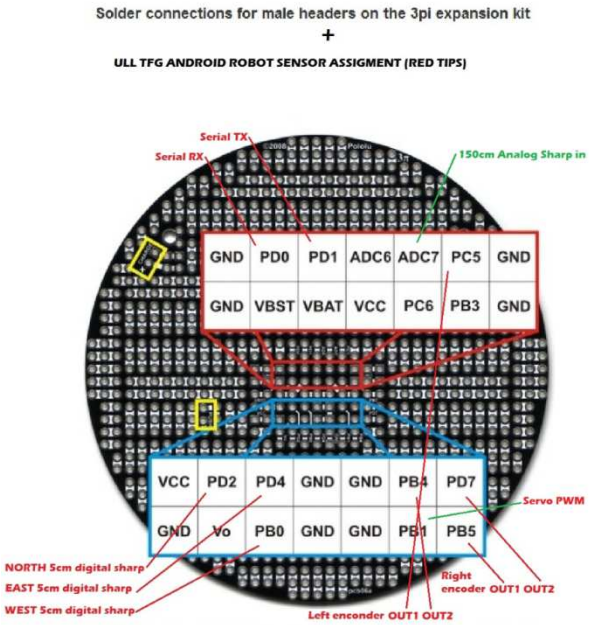
Capítulo 6

6. INSTALACIÓN DE ENCODERS Y SENSORES

Tabla 1

Función	Pin en Arduino	Pin en Mega168
I/O digital libres (x3) (Extraer el jumper PC5 para liberar el pin 19)	Pines digitales 0, 1, y 19	PD0, PD1, PC5
Entradas analógicas libres (Extrayendo los jumpers, x3)	Entradas analógicas 5-7	PC5, ADC6. ADC7
Control del motor 1 (izquierdo, A y B)	Pines digitales 5 y 6	PD5 y PD6
Control del motor 2 (derecho, A y B)	Pines digitales 3 y 11	PD3 y PB3
Sensores de reflectancia QTR,RC (de izq. A der. X5)	Pines digitales 14-18	PC0-PC4
LED rojo (izquierdo) para el usuario	Pin digital 1	PD1
LED verde (derecho) para el ususario	Pin digital 7	PD7
Botones para el usuario (izq. A der. X3)	Pines digitales 9, 12, y 13	PB1, PB4, y PB5
Zumbador	Pin digital 10	PB2
Controles del LCD (RS, R/W, E)	Pines digitales 2, 8, y 4	PD2, PB0, y PD4
Datos del LCD (4-bit: DB4 - DB7)	Pines digitales 9, 12, 13, y 7	PB1, PB4, PB5, y PD7
Control del sensor de reflectancia LED Infrarrojo (Asignarle LOW para apagarlo)	Pin digital 19 A través de jumper	PC5
Potenciómetro para el usuario	Pin digital 7 A través de jumper	ADC7
2/3 de voltaje de la batería	Pin digital 6 A través de jumper	ADC6
Líneas de programación ISP (x3)	Pines digitales 11, 12, y 13	PB3, PB4, PB5
Botón reset	reset	PC6
UART (RX y TX)	Pines digitales 0 y 1	PD0 y PD1
12C/TWI	Inaccesible	
SPI	Inaccesible	

Tabla 2



De la tabla 1 superior se extraen los pines disponibles para su uso. De los cuales serán necesarios 8 digitales, pues necesitamos 2 por encoder, 4 en total para ellos ya que cada encoder funciona utilizando en conjunto los flancos de dos sensores de reflectancia, ofreciendo combinaciones de dos bits (11 00 10 01), que dependiendo del orden en que se obtengan, es posible detectar el sentido de giro de la rueda. Necesitaremos además los tres pines para los sensores Sharp de 5cm, siendo en este caso un solo bit de sensado por cada uno de ellos, y por ultimo un PWM para el servo, el cual también será un pin digital, con la característica de que debe de tener la capacidad de funcionar como emisor de señales PWM. Dicha señal está disponible en 6 pines el "Atmega328" (*véase la referencia al "3pi" en la sección de materiales y recursos*). De la tabla también se extrae que los pines PD0 y PD1 son dedicados a la comunicación serial, (líneas RX y TX) por lo que han de prepararse dos vías cableadas para su uso como puente de comunicaciones.

Como no va a ser usada la pantalla LCD del "3pi", existen 7 pines en la tabla superior que están dedicados a dicha tabla, los cuales estarán libres para su uso una vez extraída la pantalla y solo sería necesario un pin digital mas para cubrir los 8 necesarios. Este el PC5, disponible una vez se extraiga el jumper correspondiente.

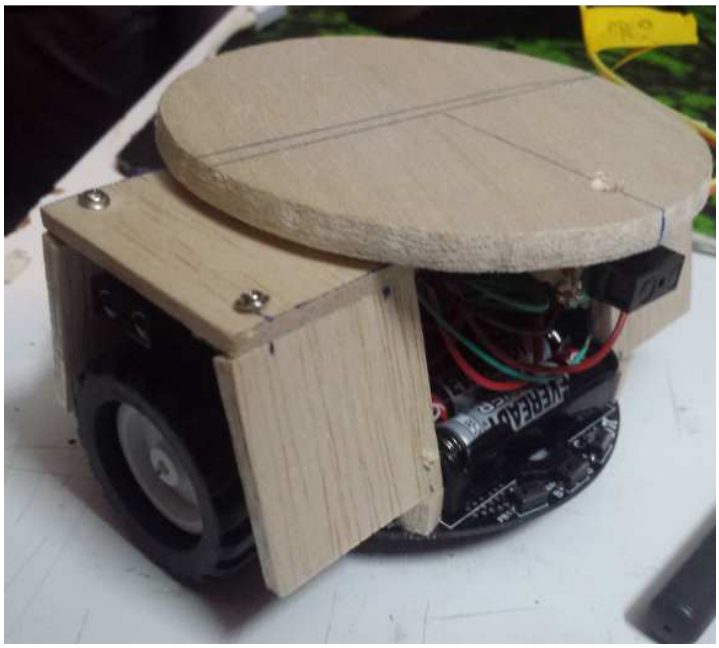
Para la señal PWM se estudian los pines de un chip "Atmega328", el cual esta también presente en el "Arduino UNO" del que se dispone, (*véase la referencia a la figura "Arduino UNO" en la sección de materiales y recursos)*, en cuya figura se observa un "~" junto a los números de los pines digitales. El cual indica que dicho pin tiene capacidad para funcionar como emisor de señales PWM, indicando que dicho pin puede ser seleccionado en el "3pi" como candidato a controlar el servo. Candidatos de los cuales observando la tabla se extrae que los pines 3, 5, 6, y 11 son utilizados para controlar los motores, así como el pin 10 está dedicado al zumbador. Esto solo deja libre el pin 9 como posible señal PWM a usar en el proyecto, por lo que debe ser reservado este pin para el servo, y asignar el resto de ellos a las 7 entradas digitales requeridas.

Una vez asignados todos los pines digitales requeridos, se procede a analizar el pin requerido para el sensor Sharp de 150cm, el cual es una entrada analógica. Dicha entrada analógica, se extrae de la tabla que se dispone de dos de ellas directamente accesibles en el conector central del "3pi" (*véase la tabla esquema de la instalación en la página anterior)*, de las cuales se selecciona la ADC7, al retirar el jumper que la puentea con el potenciómetro, que no será utilizado en el proyecto.

6.1. UBICACIÓN ÓPTIMA DE LOS SHARP DE 5CM

A la hora de ubicar los tres Sharp de 5cm existe un problema (*véase la configuración de dichos sensores en la primera figura inferior)*, y es que a la hora de probar el robot, se observa que su tasa de choques con objetos es demasiado alta, es decir que el robot en su navegación aleatoria por entornos caseros donde existen objetos lo suficientemente estrechos como puede ser la pata de una silla o de una mesa, suele chocar deteniendo su marcha de manera anómala. Comportamiento que ha sido observado gracias a un código de prueba en forma de agente reactivo simple, (*véase la referencia al código en la segunda figura, en la página siguiente)*, el cual si no detecta nada en los 3 Sharp, simplemente avanza en línea recta, hasta que detecta un obstáculo en alguno de sus sensores. Si el objeto detectado esta en Sharp de la izquierda, se corrige la trayectoria haciendo un breve giro de unos 20 grados hacia la derecha, este comportamiento se imita en el caso de detectar el obstáculo en el sensor de la derecha, corrigiendo la trayectoria hacia la izquierda en este caso. Por último si el obstáculo aparece en el sensor delantero, el robot retrocede una distancia corta de algunos centímetros para luego hacer un giro de 90º hacia la izquierda o derecha de manera aleatoria.

La configuración del código de prueba para los sensores ha sido la de un agente reactivo simple, cuando no detecta nada simplemente avanza, si detecta algo en el sensor izquierdo, gira ligeramente a la derecha mientras avanza, si detecta algo a la derecha gira ligeramente a la izquierda. Si detecta algo de frente, ira marcha atrás algunos centímetros y girara aleatoriamente 90º a la izquierda o a la derecha.

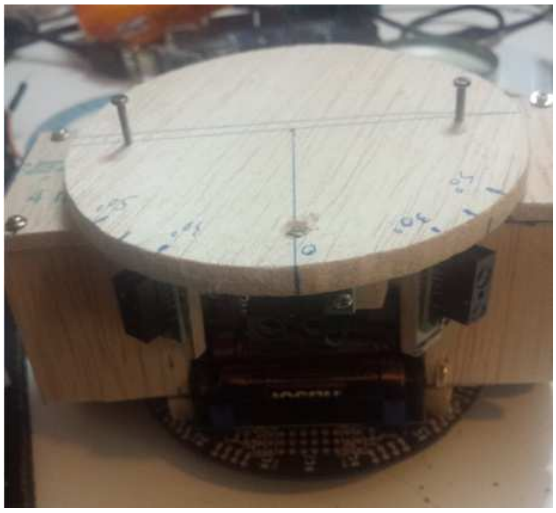


Colocación de sensores Sharp de 5cm

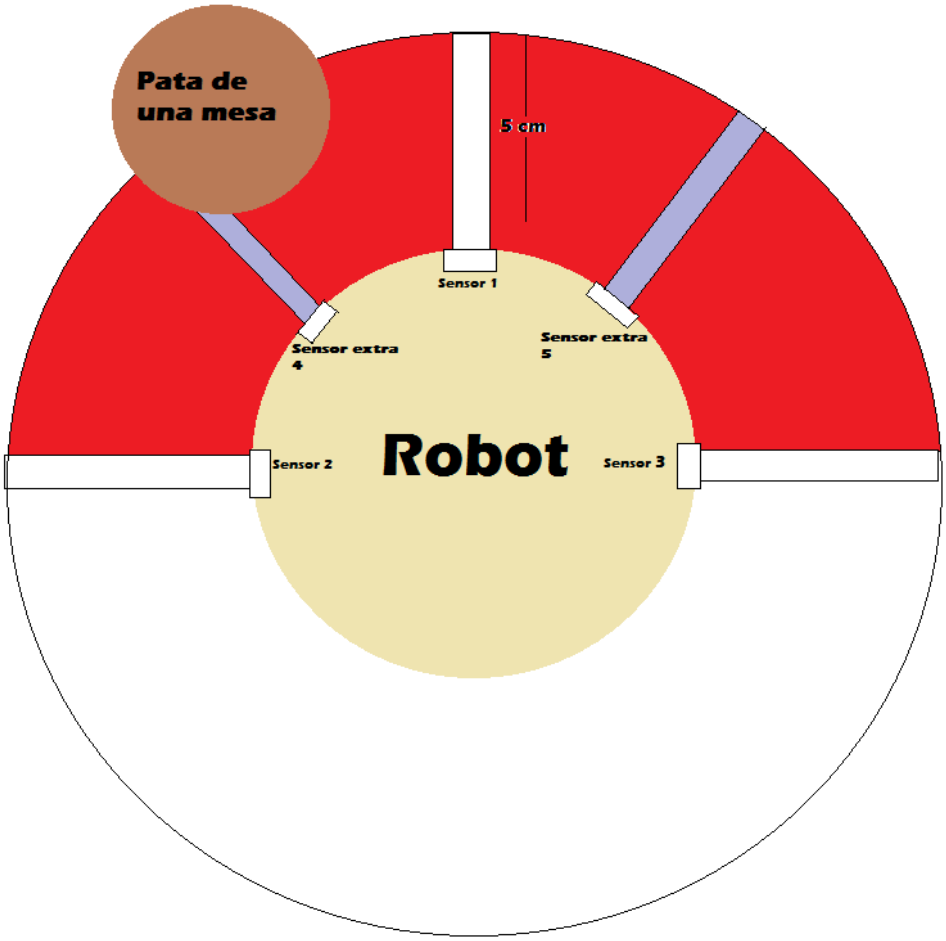
```
void setup() {
  Serial.begin(9600);
  encoders.init(enc1A, enc1B, enc2A, enc2B);
  pinMode(proxW, INPUT);
  pinMode(proxN, INPUT);
  pinMode(proxE, INPUT);

  OrangutanPushbuttons::waitForRelease(BUTTON_B);
}

void loop() {
  randnum = random(2);
  obstaculoN = digitalRead(proxN);
  obstaculoW = digitalRead(proxW);
  obstaculoE = digitalRead(proxE);
  if ((obstaculoW == LOW)&&(obstaculoE == LOW)) {
    OrangutanMotors::setSpeeds(0, 0);
    delay(50);
    OrangutanMotors::setSpeeds(-100, -100);
    delay(300);
  } else if (obstaculoN == LOW) {
    OrangutanMotors::setSpeeds(-150, -150);
    delay(300);
    if (randnum == 0) {
      OrangutanMotors::setSpeeds(-130, 130);
    } else {
      OrangutanMotors::setSpeeds(130, -130);
    }
    delay(150);
  } else if (obstaculoW == LOW) {
      OrangutanMotors::setSpeeds(80, 40);
      delay(120);
  } else if (obstaculoE == LOW) {
      OrangutanMotors::setSpeeds(40, 80);
      delay(120);
  } else {
    OrangutanMotors::setSpeeds(80, 80);
  }
}
```

Después de hacer varios ensayos con el "3pi" funcionando de esta manera se observa que la tasa de choques es demasiado alta, tiene muchos puntos ciegos debido a que los Sharp solo avanzan en línea recta y cualquier obstáculo lo suficientemente pequeño es suficiente para atascar al robot, (*véase la referencia a los puntos ciegos en la segunda figura de la página siguiente*). Perspectiva ante la cual diseñamos una posición alternativa para los sensores, (*véase la referencia a la primera de las figuras de la página siguiente*), llegando a la conclusión de que esta situación quedaría corregida en gran medida con la incorporación de dos sensores Sharp de 5cm mas en las ubicaciones señalas, a 45º de los otros 3 sensores originales, reduciendo la tasa de choques del robot en casi su totalidad. Conclusion a la que se llega gracias al estudio de los puntos muertos de los sensores realizado con la figura anteriormente referenciada.

Por desgracia, no se dispone de estos dos sensores extra, y la solución queda propuesta como posible mejora de cara al futuro, adoptando en esta ocasión una solución alternativa y menos eficiente, la cual consiste en detectar cuando el robot se ha detenido por una colisión mediante los encoders, los cuales dejarán de contar cuando el robot se detiene.



Colocación alternativa de los encoders



Estudio de los puntos muertos

La zona roja en la figura son los puntos muertos en la visión del robot mediante los Sharp. Se puede observar que añadiendo dos sensores extra la situación mejora bastante y resulta más complicado que objetos pequeños colisionen con el robot.

6.2. COLOCACIÓN DEL SHARP DE 150CM

Este sensor pretende ser utilizado para tareas de mapeado. La idea inicial consistía en que el 3pi tenía que dirigir su parte frontal a la zona que se quería explorar, ejecutando exploraciones en las que tenía que detenerse, hacer giros etc...

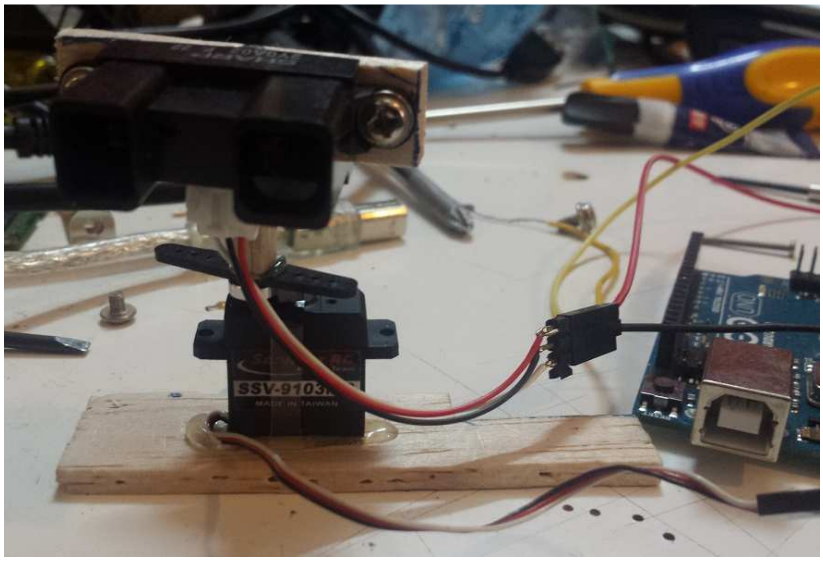
Para mejorar esta situación, se diseña un sistema para ampliar este limitado funcionamiento del sensor. Sistema que consiste en imitar el radio de acción de los sistemas de telemetría más grandes, como pueden ser los telémetros laser que actúan no solo en línea recta como el sensor Sharp sino que son capaces de detectar objetos en un gran rango de visión. El sistema diseñado incluye la colocación de un servo en cuyo eje se monta el sensor Sharp (*véase la referencia a la figura inferior)*, haciendo que éste haga barridos de 160º, enviando no solo la distancia en línea recta con los obstáculos sino un array de distancias en los 160º que abarca el servo.

El rango de acción se ha reducido a 160º debido a que el servo utilizado no llega a los 180º, siendo su máxima amplitud unos 170º aproximadamente, por lo que se toma la decisión de diseñar el sistema para que la amplitud analizada sea de 160º efectivos, empezando en 10 grados sobre la línea recta que une ambos extremos, izquierdo y derecho de la base pasando por el centro de la misma, hasta cubrir un ángulo de 160º dejando otros 10º con respecto a dicho diámetro .

El modo de funcionamiento es el siguiente, se inicializa el servo en 0 grados y se comienza la rutina siguiente:

1. Leer del Sharp,
2. calcular distancia mediante una función (*véase referencia al código en la figura más adelante)*
3. aumentar (o disminuir dependiendo de si hemos llegado a los 160º) en uno la posición del servo.

Hay que tener un cuidado extra a la hora de codificar este sistema, y es que si utilizamos la librería servo de Arduino, la amplitud máxima del servo para arduino es de 180, la cual no se corresponde con la amplitud real del servo disponible, que es de 170º. Como se quiere conseguir una amplitud máxima de 160º se prueban los valores reales mediante ensayos con el servo, y se llega a la conclusión de que para un valor de 170, el servo describe los 160º deseados, con lo cual solo se necesita restar los 10 valores sobrantes para obtener una medida fiable para nuestro array de distancias. Valores que han sido eliminados de manera uniforme en los 170, ignorando una posición cada 17 valores.



Conjunto servo-Sharp

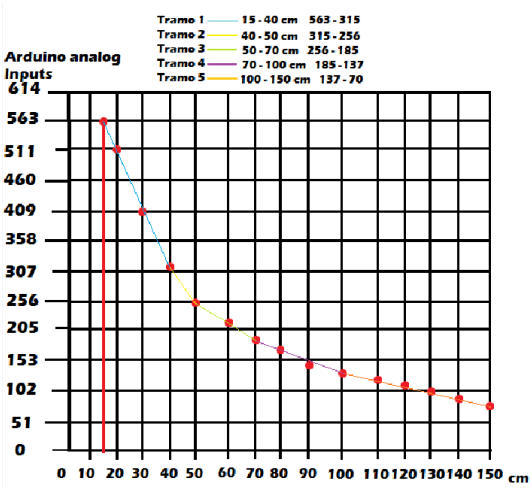
6.3. CALCULAR DISTANCIAS CON EL SHARP DE 150 CM.

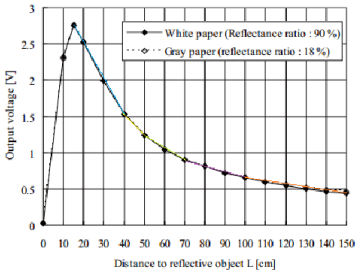
Este sensor ofrece una entrada analógica en el rango de aproximadamente 0,4V para las distancias más largas hasta 2,75V para unos 14cm. Estas medidas traducidas a los valores analógicos en Arduino, vienen a ser unos 563 para los 150cm, y 70 para los 14cm.

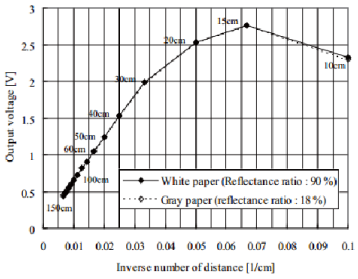
El problema es que la función que relaciona los centímetros medidos con los valores de voltaje no es lineal. Es decir que no evoluciona de manera proporcional en toda la función sino que describe una curva que ni siquiera es uniforme, pareciéndose mas a una función logarítmica, la cual por fortuna se aproxima en varios tramos a líneas rectas, las cuales serán usadas para dividir la grafica en tramos que permitan extraer la distancia con una precisión más aceptable (*véase las tablas 3, expuestas en la página siguiente)*. Dichos tramos, van a ser 5 en total y dependiendo de en cuál de ellos caiga la medición, se calculará de una manera u otra la distancia.

En el código, se modela una función, dividida en 4 operaciones que incluyen los valores mínimos y máximos que puede adoptar la señal en cada tramo, devolviendo como resultado una distancia pasada a centímetros, la cual será almacenada en un array después de cada barrido y pasada por serial al dispositivo Android para su procesamiento e inclusión en sus funciones de mapeado.

Tabla3, Tabla3.1, Tabla3.2








```
void mide() {
  double acotada;
  double bruto;
  //150cm es el maximo, 14cm es el minimo
  //la funcion no es lineal, asi que dividimos por tramos
  sensorVA0 = analogRead(sensorA0);
  if (sensorVA0 < 70)
    sensorVA0 = 70;
  if (sensorVA0 > 314) { //tramo superior 14 - 40 cm 545 - 315
    acotada = sensorVA0 - 315;
    bruto = ((26 * acotada) / 230);
    resultado = bruto - 26;
    if (resultado < 0)
      resultado = (resultado * -1);
    resultado = resultado + 14;
  } else if (sensorVA0 > 255) { //tramo 2° 40 - 50 cm 315 - 256
    acotada = sensorVA0 - 256;
    bruto = ((10 * acotada) / 59);
    resultado = bruto - 10;
    if (resultado < 0)
      resultado = (resultado * -1);
    resultado = resultado + 40;
  } else if (sensorVA0 > 184) { //tramo 3° 50 - 70 cm 256 - 185
    acotada = sensorVA0 - 185;
    bruto = ((20 * acotada) / 71);
    resultado = bruto - 20;
    if (resultado < 0)
      resultado = (resultado * -1);
    resultado = resultado + 50;
   } else if (sensorVA0 > 136) { //tramo 4° 70 - 100 cm 185 - 137
    acotada = sensorVA0 - 137;
    bruto = ((30 * acotada) / 48);
    resultado = bruto - 30;
    if (resultado < 0)
      resultado = (resultado * -1);
    resultado = resultado + 70;
   } else { //tramo inferior 100 - 150 cm 137 - 70
    acotada = sensorVA0 - 70;
    bruto = ((50 * acotada) / 67);
    resultado = bruto - 50;
    if (resultado < 0)
      resultado = (resultado * -1);
    resultado = resultado + 100;
   }
}
```

Código de traducción a medidas en cm

Capítulo 7

7. EL PROTOCOLO DE COMUNICACIÓN SERIAL

Es necesario el diseño de este protocolo de comunicación para que el robot sea capaz de mantener una comunicación solida y estructurada con el dispositivo receptor, en este caso el teléfono con sistema operativo Android. Para dicho diseño, se ha decidido usar como modelo el protocolo usado por el robot PIONEER del que dispone la ULL.

Este protocolo se basa en enviar paquetes con el siguiente formato, siempre en palabras de 2 bytes de longitud, enviando en primer lugar dos palabras de sincronización, la 0xFA y 0xFB. Estas dos palabras han de venir una detrás de la otra, en caso de recibir la 0xFA se pasa a esperar la 0xFB, y en caso de no coincidir con ésta lo que se recibe, se comprueba que no se trata de otro 0xFA en cuyo caso habría que seguir esperando por el 0xFB. Una vez recibidas se especifica el tamaño en palabras del bloque que viene a continuación, donde se espera el opcode, los argumentos y el checksum.

7.1. EL PAQUETE DE DATOS:

La primera palabra de este paquete de datos indica el numero de palabras del paquete, es lo que es llamado "Lenght" en el código mostrado más adelante, recibiendo a continuación el opcode de la instrucción, el cual será único para cada una de las posibles operaciones a utilizar en el robot. Este opcode determina el número y tipo de argumentos que han de venir a continuación, los cuales constan de 2 palabras para un entero, cadenas de palabras representando símbolos ASCII terminadas en null '\0' y cadenas de longitud indeterminada de palabras para otros datos.

7.2. EL CHECKSUM

El checksum es un calculo que se realiza para comprobar que un paquete de datos no está corrupto, y que se corresponde con paquete que salió del dispositivo origen, obligando a su cálculo a ambos lados de la comunicación, una vez para su envío, y otra para comprobar la integridad a la hora de recibir los datos.

Este cálculo se realiza analizando palabra a palabra el paquete y guardando el resultado en dos palabras que serán enviadas al final de el paquete, volviendo a realizar el calculo en el sistema destino para comprobar igualdad, (*véase la figura del código en la pagina siguiente).*

```
int calcChecksum() {
  unsigned int c = 0;
  int n = lenght - 2;
  int posicion = 0;

  while (n > 1) {
    c = c + ((pack[posicion]<<8) | (pack[(posicion + 1)]));
    n = n - 2;
    posicion = posicion + 2;
  }
  if (n > 0)
    c = c ^ pack[(posicion + 1)];

  return(c);
}
```

Código para el cálculo del checksum

7.3. IMPLEMENTACIÓN:

Para la implementación del protocolo, se ha seguido una estructura en forma de autómata. Es decir, el programa pasa a un estado nuevo cada vez que se consigue una parte nueva del paquete y así se lleva un control del flujo de ejecución (*véase la referencia al código en la página siguiente)*. Éste autómata empieza en el estado 0, esperando un 0xFA, si lo encuentra pasara al estado 1, sino volverá al estado inicial. En el estado uno, si le llega un 0xFB pasara al estado 2, si le llega otro 0xFA seguirá en el estado 1 y si le llega otra cosa, volverá al estado 0.

El autómata volverá al estado 0 si se rompe el flujo de ejecución, esperando un nuevo comando para avanzar por su estructura, llegando a procesar un comando solo en el caso de que alcance el ultimo estado, en el que se calcula el checksum y se ejecuta el comando si se comprueba el cálculo.

En la implementación, a medida que se reciben los datos se componen las palabras de 16 bits concatenando dos bytes que son recibidos por separado por el "3pi", debido a que solo puede leer datos byte a byte, problema que se soluciona multiplicando la primera lectura por 256, y sumándole la segunda lectura.

```
void loop() {
  if(Serial.available() >= 10) {
    Serial.println(Serial.available());
    for (int i = 0; i < Serial.available(); i++) {
      entrada = Serial.read() * 256;
      entrada = entrada + Serial.read();
      switch (estado) {
        case 0:
          Serial.println("case 0 necesita 0xFA");
          Serial.println("proporcionado");
          Serial.println(entrada);
          Serial.println("fin");
          if (entrada == 0xFA)
            estado = 1;
          break;
        case 1:
          Serial.println("case 1 necesita 0xFB");
          if (entrada == 0xFB) {
            estado = 2;
          } else if (entrada == 0xFA) {
            estado = 1;
          }
          break;
        case 2:
          Serial.println("case 2 lenght");
          lenght = entrada;
          estado = 3;
          break;
        case 3:
          Serial.println("case 3 pack");
          if (contador < (lenght - 3)) { //lenght - 2 es el tamaño del pack, -3 por empezar en 0
            pack[contador] = entrada;
            contador++;
            estado = 3;
          } else {
            pack[contador] = entrada;
            contador = 0;
            estado = 4;
          }
          break;
        case 4:
          Serial.println("case 4 checksums");
          if (contador == 0) {
            checksum1 = entrada;
            contador++;
            estado = 4;
          } else {
            checksum2 = entrada;
            contador = 0;
            muestracepta();
          }
          break;
      }
    }
}
```

Codificación del autómata para la comunicación serial

Capitulo 8

8. PARTE DEL DISPOSITIVO ANDROID.

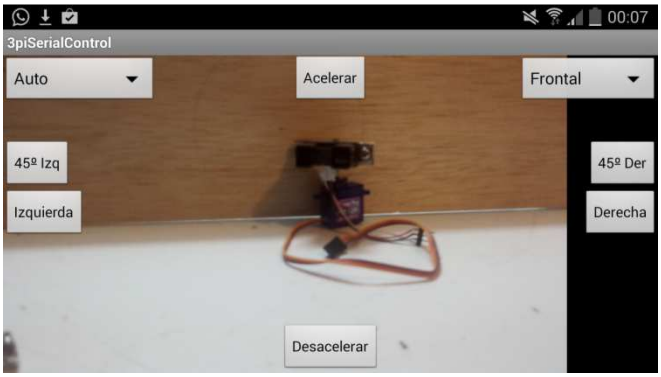
8.1. PLATAFORMA.

Se construye una plataforma nuevamente gracias a la madera de balsa, la cual es forrado con terciopelo para acomodar suavemente el dispositivo que se vaya a colocar (*Véase la figura inferior)*, consiguiendo así un aspecto para el robot mucho más serio, y "actual".

Estructura definitiva

8.2. DESARROLLO DE LA APLICACIÓN "CEREBRO".

Se desarrolla una aplicación android, que como pantalla principal muestre la cámara del teléfono, y además disponga de unos controles para moverlo. (*véase la figura de la página siguiente).*



Interfaz de la aplicación Android

Mediante dicha aplicación se pueden ejecutar algunos comandos directamente en el móvil y el "3pi" respondería a las ordenes, así como permitiéndole también la obtención de datos acerca del estado del robot, así como recibir los arrays con los valores de las distancias para el mapeado.

Para lograr la comunicación serial en el código de la aplicación, se ha utilizado una librería en java llamada "Usb serial for android", librería de código libre publicada en google code. El repositorio principal de la aplicación estaba desactualizado, así que después de indagar a cerca de porque no funcionaba la comunicación se da con el problema de que en la versión publicada no se incluía el soporte para el adaptador "CP2102 Usb to Serial".

Afortunadamente se encuentra un parche publicado por el mismo autor de la librería que corregía el problema, aunque no lo ofrecía como una librería compilada, así que se ha tenido que añadir manualmente al proyecto

8.3. EL AUTÉNTICO REMOTO

El teléfono Android puede ser directamente controlado con algún programa de control remoto como puede ser el Teamviewer, aplicación gratuita para control remoto de PCS.

Sin embargo esta solución tiene el problema de que obtenemos una latencia generalmente alta y las imágenes de la cámara llegan con bastante retraso.

Por ello, se ha decidido desarrollar una aplicación en java, utilizando la tecnología de sockets para interactuar con el 3pi de manera remota dejando la función de la cámara a programas ya existentes de código libre para el streaming de la cámara hacia el remoto.

Conclusiones y trabajos futuros

9. CONCLUSIONES

Como conclusión, se han afrontado problemas muy comunes en la robótica móvil como es el mapeado y el posicionamiento. Los cuales contribuyen a aumentar en gran medida la experiencia en este campo.

Se ha llegado a la conclusión de que es posible realizar proyectos interesantes con herramientas de código libre como las utilizadas y materiales de bajo coste. Podría decirse que este robot podría ampliarse hasta conseguir imitar todo lo que hace el robot PIONEER utilizando materiales de bajo coste y hardware y software libre. Siguiendo esta filosofía creamos un repositorio en github donde se puede acceder a todo el código del proyecto:

https://github.com/etsiiull/3pidroid

Ha sido posible comprobar cómo se trabaja en el taller del departamento de Ingeniería de Sistemas y Automática y Arquitectura de Computadores de la mano del tutor Jesús Torres, quien ha asistido y supervisado en labores tales como el soldado de componentes y la puesta en marcha de algunas funcionalidades.

Por último resaltar la oportunidad de mostrar el trabajo en la primera feria tecnológica de Tegueste junto al grupo de robótica de la universidad de la laguna y su extraordinario Verdino en el cual están aplicados todos los conocimientos utilizados en este proyecto de manera mucho más avanzada y efectiva.

10. TRABAJOS FUTUROS

Una propuesta de ampliación para este proyecto que se podría llevar a cabo es la de sustituir el "3pi" por una placa diseñada para tal fin, utilizando un ATMEGA 328 compatible con el zócalo de una placa "Arduino UNO" tradicional. Esta alternativa sería muy interesante de llevar a cabo con un coste casi nulo de componentes.

Otro trabajo futuro es el de mejorar la parte del control remoto entre un PC y el Android. Pero lo cierto es que este tema es tan amplio que podría hacerse todo un nuevo proyecto con solo este aspecto.

Summary and conclusions.

As conclusion, We had to face several common problems on mobile robotics, as mapping is. That really helps to get much more experience in this topics.

Other conclusion is that it is possible to make interesting projects with low cost and open source materials as the ones used. This project can be evolved to the point of being an alternative to the PIONEER robot, using open source hardware and software. Following this philosophy we decided to create a repository where all the code developed in this project is:

https://github.com/etsiiull/3pidroid

It was an opportunity to see how the work is done in the workshop of the Systems Engineering and Automatic and Computer´s Architecture Department (ISAATC for its Spanish initials).

Finally, it's worth to highlight that the work could be shown on the first technology fair of Tegueste with the group of robotics of the ULL and their great robot Verdino, which has all the knowledge applied in this project, but in a way more advanced and professional than this project.

Presupuesto

Este es el presupuesto general del proyecto, incluye solo los costes de material utilizado:

Tabla 4

Nº	Articulo	Precio	Cantidad
1	Pololu M3pi	110.31€	1
2	Placa MBED	36.04€	1
3	Arduino UNO	20 €	1
4	Programador USBAsp	3 €	1
5	Puente USB-Serial CP2102	3 €	1
6	Madera de balsa (lámina 1m)	4.5€	2
7	Sensor Sharp de 5cm	5.10€	3
8	Pololu Wheel encoder	10.98€	2
9	Pareja de ruedas 42x19mm	5.13€	1
10	Sensor Sharp de 150cm	10.25€	1
11	Servo TowerPro SG92R	8 €	1
	TOTAL	241.99€	

Bibliografía

Repositorio del proyecto:
https://github.com/etsiiull/3pidroid
Pololu:
http://www.pololu.com/
Manual de usuario de 3pi:
http://www.pololu.com/docs/0J21
Arduino:
http://www.arduino.cc/es/
Nasa Curiosity:
http://www.nasa.gov/mission_pages/msl/index.html
"Botiful":
http://www.botiful.me/
Wiring:
http://www.wiring.org.co/
Android:
http://www.android.com/
MBED:
https://mbed.org/
Teamviewer:
http://www.teamviewer.com/es/Index.Aspx
Java:
http://java.com/es/