EL ROBOT BUTIÁ

Temas

- Aspectos generales del BUTIÁ
- Módulos, drivers y sistema Plug & Play, Api

Aspectos generales del BUTIÁ

- Diseño mecánico
- Hardware
- Software
- Api
- Extensión

Diseño mecánico

- Chasis constructivo
- Ruedas
- Amortiguador
- Motores

Chasis constructivo

- Hecho en acrílico
  - Cortes en láser
- Superficie suficiente para llevar un laptop o XO.
- “Agujeritos” para poder poner sensores o actuadores (constructivo).
- Barandas para proteger la XO.

Ruedas

El robot cuenta con dos pares de ruedas

- Dos ruedas de acrílico las cuales están sujetas a los motores.
- Dos ruedas locas, una de ellas amortiguada para poder sortear desniveles.

Amortiguador

Para poder evitar desniveles el robot tiene una placa amortiguadora hecha en acero de alto carbono de 0.5 mm. de espesor.

Una de las “ruedas locas” está sujeta a dicha placa.

Motores

- Motores de continua.
- Precisión 1/3 de grado.
- 12kg.cm (7V), 16,5kg.cm (10V).
- Costo U$S 40.
- Se pueden poner en cascada conectados al mismo bus.
- Fuertes, fácil de controlar. Buena relación precio/torque.
- Tienen cierta “inteligencia”.

Hardware

- Componenes de hardware:
  - Placa E/S basada en un microcontrolador.
  - Shield.
  - XO o SBC.
  - Sensores/Actuadores

Placa E/S basada en microcontrolador

- Dado que la XO no tiene puertos de e/s digital/analógicos generales con los cuales leer sensores necesitabamos algo que nos facilite la interacción con los mismos.
- La opción elegida fue una SBM (Single Board Microcontroler) que tiene muchos puertos (A/D) y nos permite manejar a nuestro "gusto" los pines.
- En este proyecto utilizamos SBMs con conexión USB para conectarse a la XO.

Placa Arduino Megas

- Características:
  - Microcontrolador Atmega1280
  - Voltaje de funcionamiento 5V
  - Pines E/S digitales 54 (14 proporcionan salida PWM)
  - Pines de entrada analógica 16
  - Memoria Flash128 KB de las cuales 4 KB las usa el gestor de arranque (bootloader)

Placa Arduino Mega

- Esta placa se encarga de gestionar la interacción con los sensores y actuadores.
- Provee un IDE para programarla en C++.
- En lugar de esta placa se puede utilizar una USB4all.

Shield

- Necesitabamos exponer los pines de manera de formar conectores.
- Ofrece cierta circuitería extra para diferentes propósito:
  - Comunicación con motores
  - Conectores para sensores/actuadores
  - Control de motores tipo steppers

Shield

- Diseño hecho por el grupo.
- Doble capa.
- Conección polarizada para 8 sensores/actuadores.
- 1 puerto para bus Dinamixel (Motores AX12).
- Incluye el integrado “L293” que nos permite controlar diferentes tipos de motores (continua, steppers).

XO/SBC

- Este dispositivo se encarga de dar soporte a la lógica de alto nivel, la cual permite interacción con cualquier programa que se pueda comunicar por “socket” (socket: permite comunicación de procesos entre diferentes dispositivos)
- A este nivel es bueno tener mayor capacidad de procesamiento ya que nuestros programas pueden llegar a ser complejos. Ejemplo: procesamiento de imágentes, redes de aprendizaje.
- Nos permite tener mayor nivel de abstracción.

XO/SBC

- Nos brinda las herramientas para desarrollar la lógica de control del robot con mayor nivel de abstracción y evita tener que colocar mucho código en el firmware disminuyendo la complejidad del mismo.
  - Permite desarrollar código con mayor grado de mantenibilidad.
  - El firmware sólo es responsable de interactuar con los dispositivos.
  - Permite mayor portabilidad del sistema.

Sensores/Actuadores

- Se conectan mediante conectores polarizados al shield.
- Existen dos tipos de sensores:
  - Analógicos
  - Digitales

Sensores Digitales

- Contacto
- Vibración
- Inclinación
- Contacto (capacitivo)

Sensores Analógicos

- Luz
- Temperatura
- Magnético
- Escala de grises

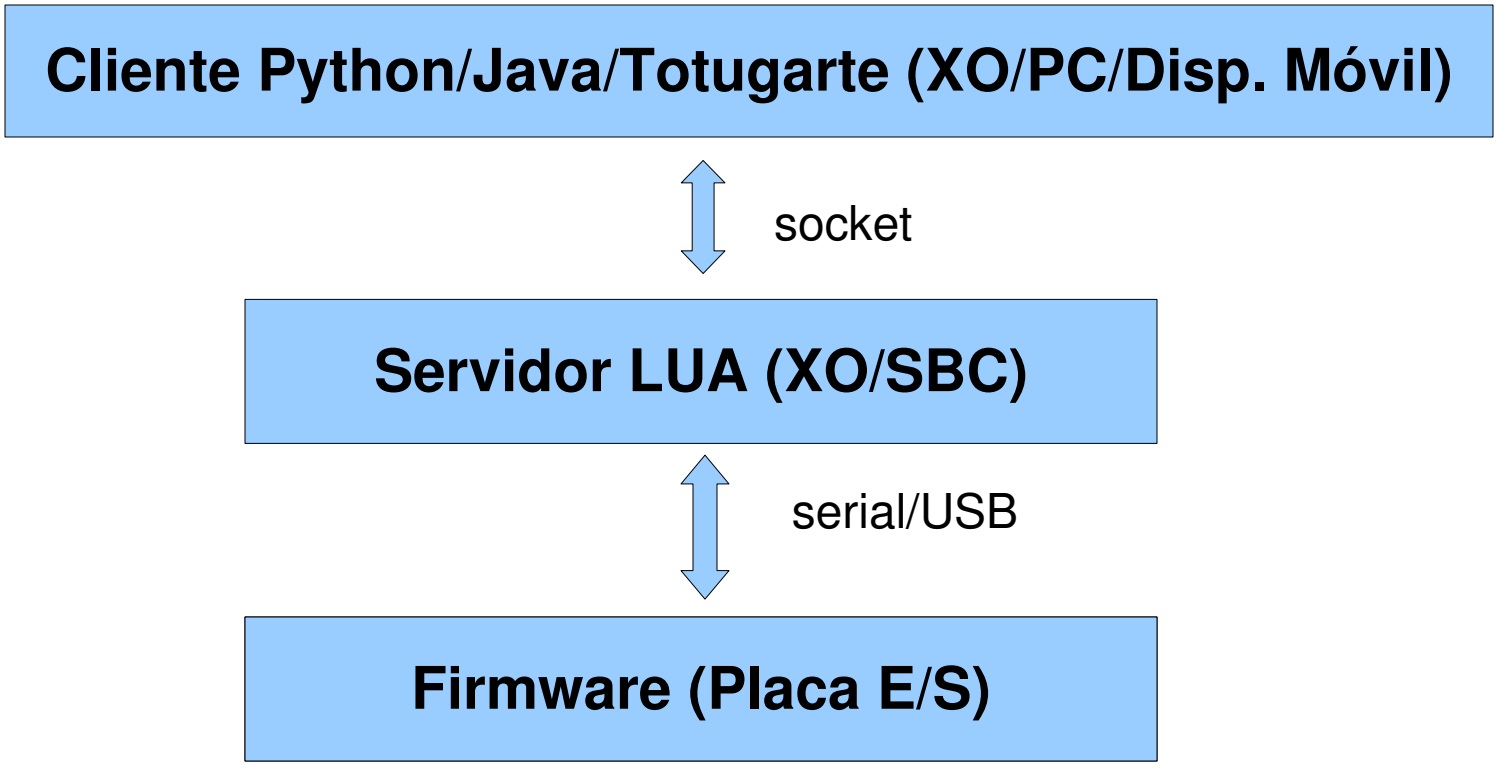
Actuadores

- Led
- Motores

Software

- Componentes:
  - Firmware (C++)
  - Biblioteca Serial (C)
  - Servidor (LUA) (Bobot- Server)
  - Api (Python, Java, Lua)
  - Cilente
- Comunicación
- Arquitectura
- Plug & Play

Comunicación y componentes



Firmware

- Desarrollado a medida para las necesidades del BUTIÁ.
- Trabaja utilizando “módulos de usuario”. Uno por cada tipo sensor/actuador, más algunos fijos.
- No es un simple “pasamanos”. Implementa lógica que por razones de performance o restricciones tiempo real es necesario hacerlas a bajo nivel.
- Especializado en la comunicación con dispositivos electrónicos.

Servidor

- Programado en LUA.
- Encapsula el comportamiento de los sensores/actuadores mediante “drivers”.
- Soporta conexión con diferentes tipos de controladoras de e/s. (Arduino, USB4all)

Butiá Api

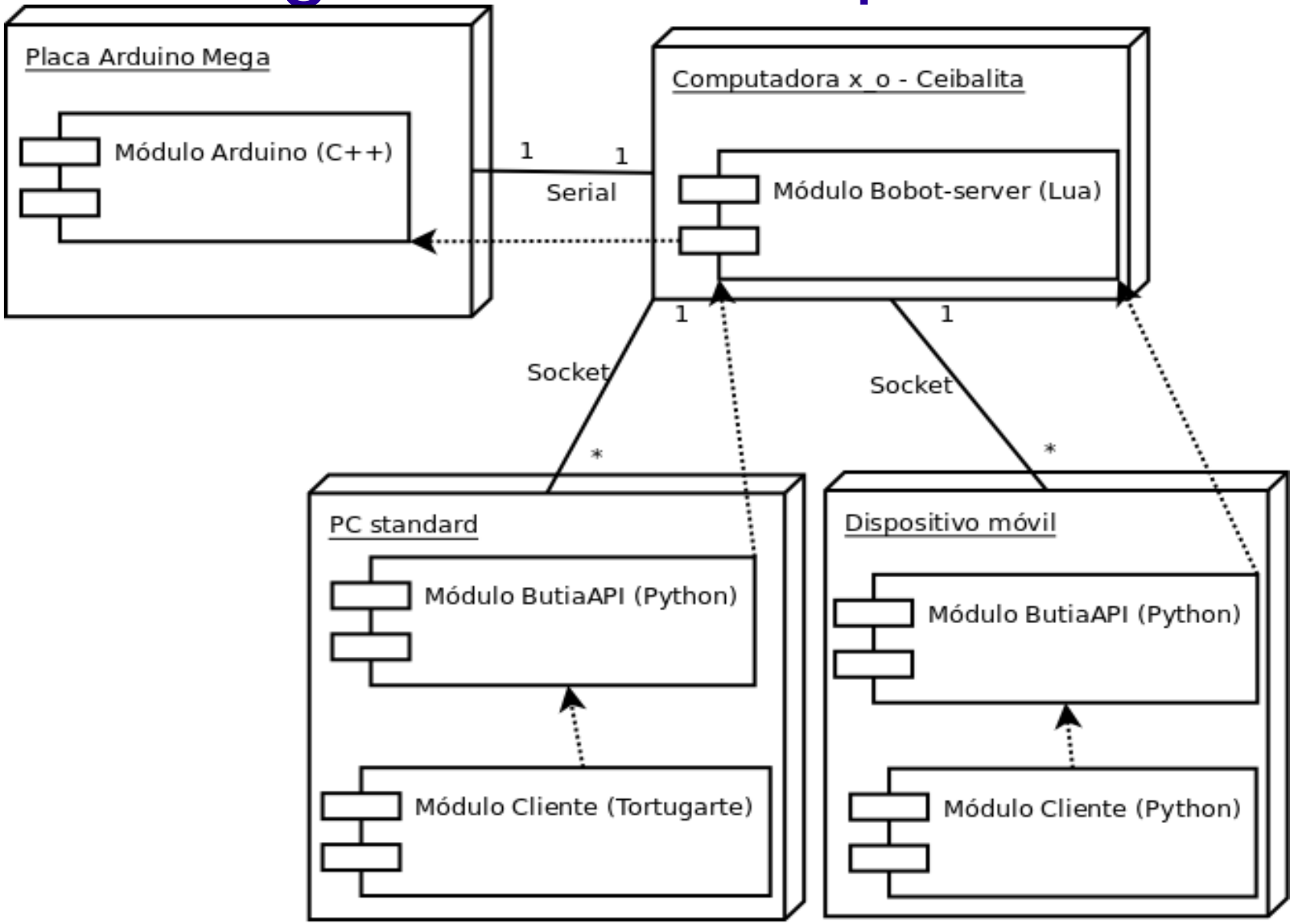
- Expone operaciones de alto nivel para obtener valores de sensores o ejecutar acciones de los actuadores.
- También nos permite consultar la carga de las baterías, listar módulos de usuario, hacer un ping, entre otras funciones.
- Progamada en Python.
- Se comunica por medio de sockets con el servidor.

Cliente

- Tortugarte.
- Programa en Python corriendo en un celular o en la misma XO o SBC.
- Chat de la XO.

```
import butiaAPI
butiabot = butiaAPI.robot()              #pido una instancia del robot
butiabot.listarModulos()                 #listo módulos de usuario
error = False
while not error:
        sen1 = butiabot.getBoton()   #leo el valor del sensor
        if sen1 == -1:
                error = True
        else:
                print sen1
butiabot.cerrar()                        #cierro la comunicación
```

Diagrama de componentes



Comunicación

- Dos protocolos diferentes:
  - Entre el Cliente y el Servidor LUA.
  - Entre el Servidor LUA y la Placa de E/S.

Comunicación Cliente - Servidor

- Orientado a Strings.
- Operaciones básicas: CALL, LIST, OPEN, CLOSE, CLOSE ALL.
- Utiliza los sockets del S.O..
- Ejemplos:

  operación módulo función [[parámetro1] .. [parámetroN]]

  "CALL motores setvel2mtr 0 500 1 200"

  “LIST”

  "CALL boton getBoton"

Comunicación Servidor LUA - Placa de E/S

- Orientado a Bytes.
- Utiliza comunicación Serial/USB.
- Escalable a nuevas tecnologías de comunicación (bluetooth, infrarojo, entre otros).

Arquitectura

- Cliente – Servidor.
- Independiente de la Placa E/S y de la Plataforma de Control.
- La Placa E/S implementa servicios que luego son descubiertos por el Servidor LUA mediante los “drivers”. Luego el Servidor expone estos servicios con un mayor nivel de abstracción

Plug & Play

- Nos permite descubrir de manera automática que sensores/actuadores están conectados.
- Intuitivo. Similar a periféricos USB.
- Sistema de codificación para un gran conjunto de dispositivo.

Plug & Play

- Ejemplo: “boton, boton1”. Si es necesario podemos saber en que conector se encuentra dicho sensor.
- Se potencia en lenguajes interpretados.
- Independiza el programa respecto del puerto al cual estén conectados los sensores/actuadores.

Extensión

- Mecánica
  - Sustituír los materiales por desecho electrónico.
    Ejemplo ruedas hechas con cds., chasis hecho en madera, motores de impresora, entre otros.
- Hardware
  - Shield para Arduino.
  - Shield para USB4all.
  - SBC.
  - Agregarle un brazo mecánico

Extensión

- Software
  - Interactuar con la roomba a través de la XO.
  - Butia/USB4All
  - Extender a Scratch.
  - Extender a EToys.
  - Extender al nuevo kit de sensores.
  - SBC/Butia
  - Biblioteca de video.
  - Agregar nuevos sensores GPL/GNC, otros.
  - Measure (graficar sensores en pantalla usando sensores de butia)

Referencias

- http://www.fing.edu.uy/inco/proyectos/butia/
- http://arduino.cc/en/Main/ArduinoBoardMega
- http://www.robotshop.ca/
- http://www.crustcrawler.com/products/bioloid/docs/AX-12.pdf

¿Preguntas?