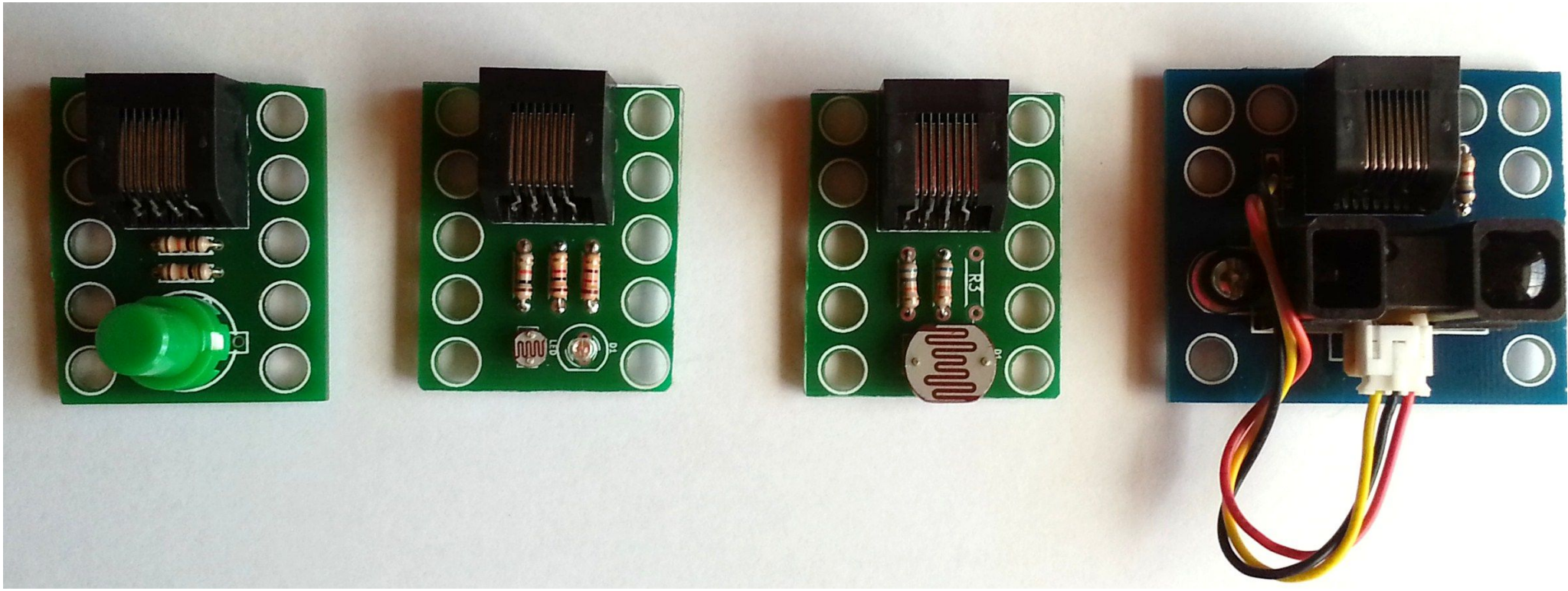
USB4Butiá + sensores utilizados en el curso

Temas

- Sensores utilizados en el curso
- USB4Butiá
- bobot
- user modules
- drivers
- bobot

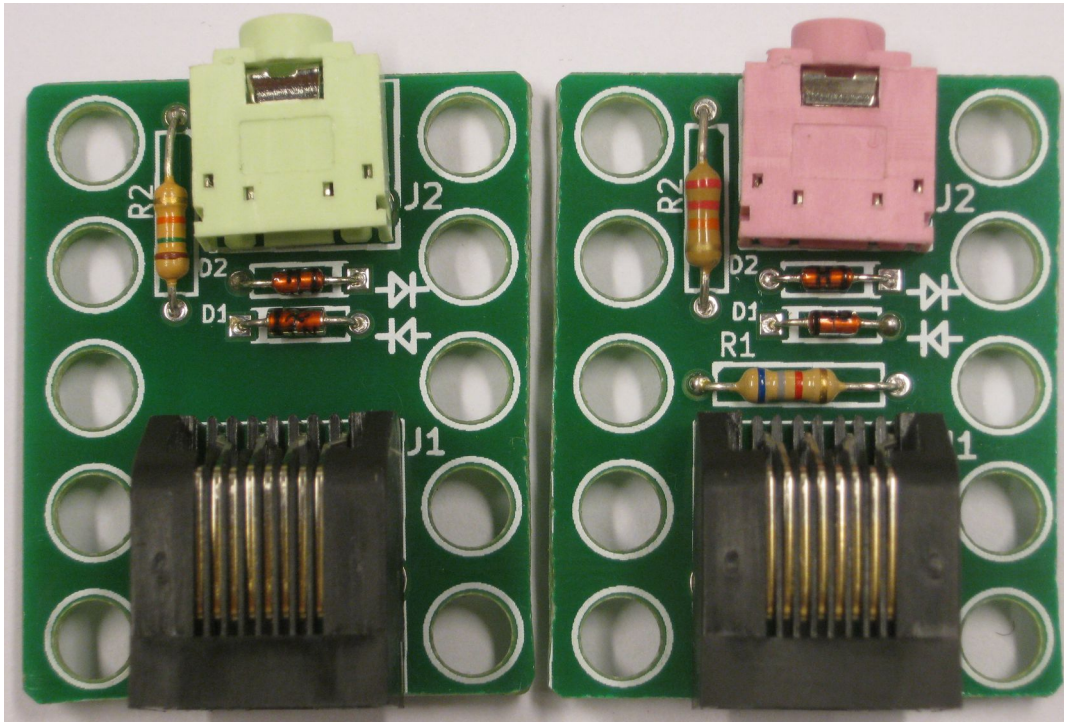
Sensores (básicos)

- Utilizaremos los siguientes sensores, de izquierda a derecha: contacto, grises, luz, distancia

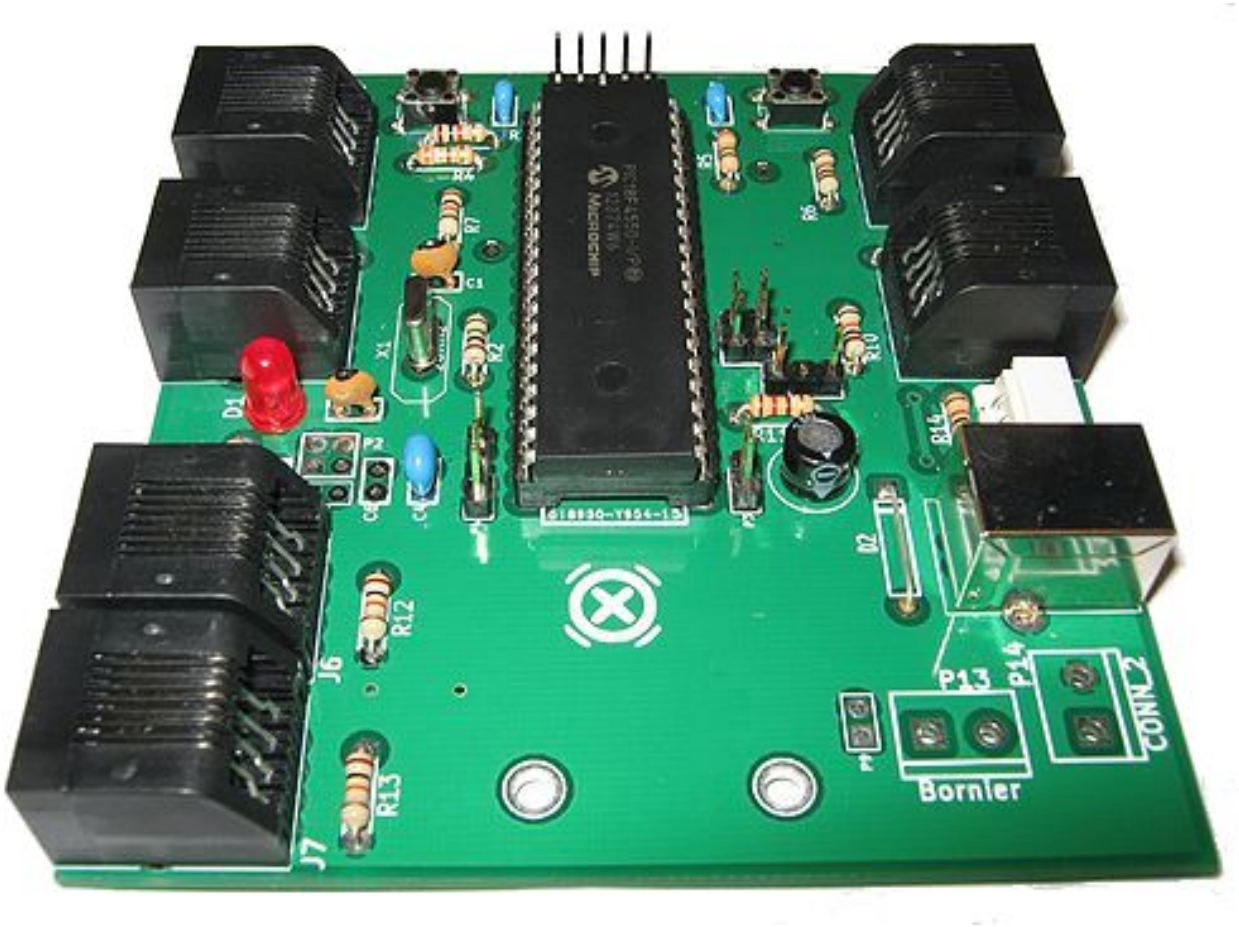


Sensores

- Para crear nuevos sensores utilizaremos módulos genéricos.
- De izquierda a derecha: sensor de voltaje, sensor de resistencia



USB4Butiá



USB4butiá

- USB4Butiá[1] es una interfaz de Entrada/Salida USB derivada de Usb4all[2]
- Especializada en el uso del robot Butiá
- Interfaz abierta y libre
  - De fácil construcción mediante mecanismos manuales
  - Fuentes públicos en git USB4all (sourceforge)
- Económica
- Fácil uso y extensión

USB4Butiá

- Construcción con materiales disponibles en el mercado local
- 6 puertos con soporte Plug&Play y Hotplug
- Bus para motores AX12
- Hack-points para acceso a bajo nivel

USB4Butiá

- Cada sensor/actuador es modelado como una entidad.
- La entidad encapsula las características y funcionamiento del sensor/actuador.
- Estas entidades se implementan como módulos en el firmware donde se interactúa a bajo nivel con el dispositivo (*user modules*).

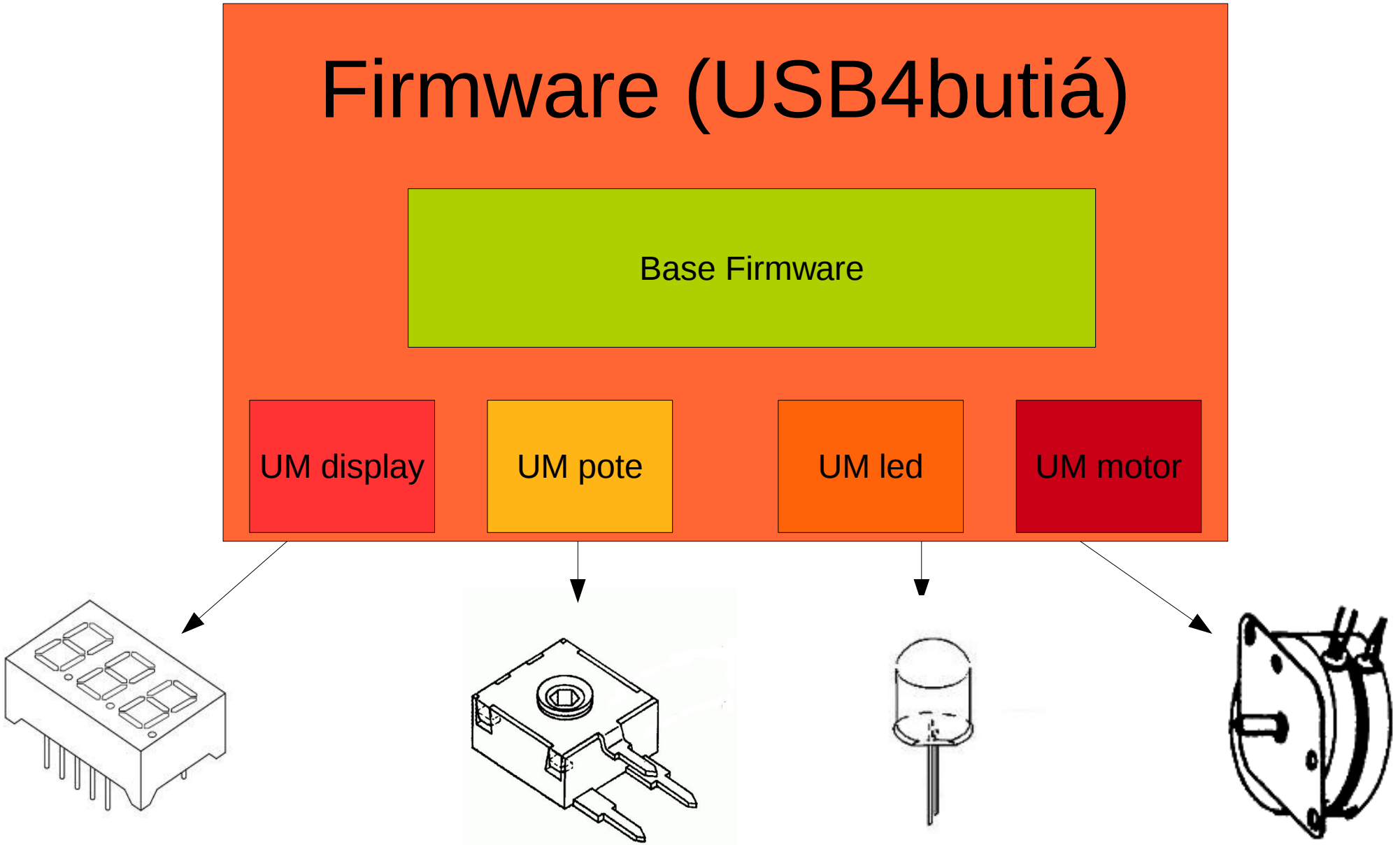
User Modules

- Componentes de software implementados en el firmware, encargados de encapsular la lógica específica para el manejo de determinado dispositivo o conjunto de dispositivos.
- El usuario sólo debe concentrarse en la lógica propia del sensor/actuador a controlar.
- Lógica independiente del mecanismo de comunicación con el PC.
- Permite extender de forma muy simple las funcionalidades del firmware.

User Modules

- Fomenta un diseño modular de la solución.
- Fomenta el reuso de los módulos para resolver diferentes problemas.
- Se identifican por su nombre (7 caracteres).
- Exponen servicios al host, por ejemplo un motor: detenerse, avanzar, retroceder.
- Fácil desarrollo. No es necesario conocer cómo se gestiona el firmware para desarrollar un módulo de usuario. Bajo impacto en el código del firmware.
- Sólo es necesario implementar algunas funciones para implementar un nuevo módulo de usuario

User Modules



User Module

Cada módulo debe definir su vector de configuración:

- Funciones CallBack para la inicialización y liberación del módulo
- Nombre del módulo

```
#pragma romdata user

const uTab userButtonModuleTable = {&UserButtonInit, &UserButtonRelease,
"button"};

#pragma code
```

La función de init se encarga de la configuración del módulo:

```
void UserButtonInit(byte handler) {

    /* add my receive function to the handler module, to be called       Función encargada de manejar
     automatically when the   pc sends data to the user module*/         la recepción de datos al módulo
button

    setHandlerReceiveFunction(handler, &UserButtonReceived);

    /* initialize the send buffer, used to send data to the PC*/

    sendBufferUsrButton = getSharedBuffer(handler);                      Solicitud de recursos para el módulo

    getPortDescriptor(handler)->change_port_direction(IN);               Configuración del puerto

}/*end UserButtonInit*/
```

User Module

La función de release se encarga de la liberación de recursos del módulo:

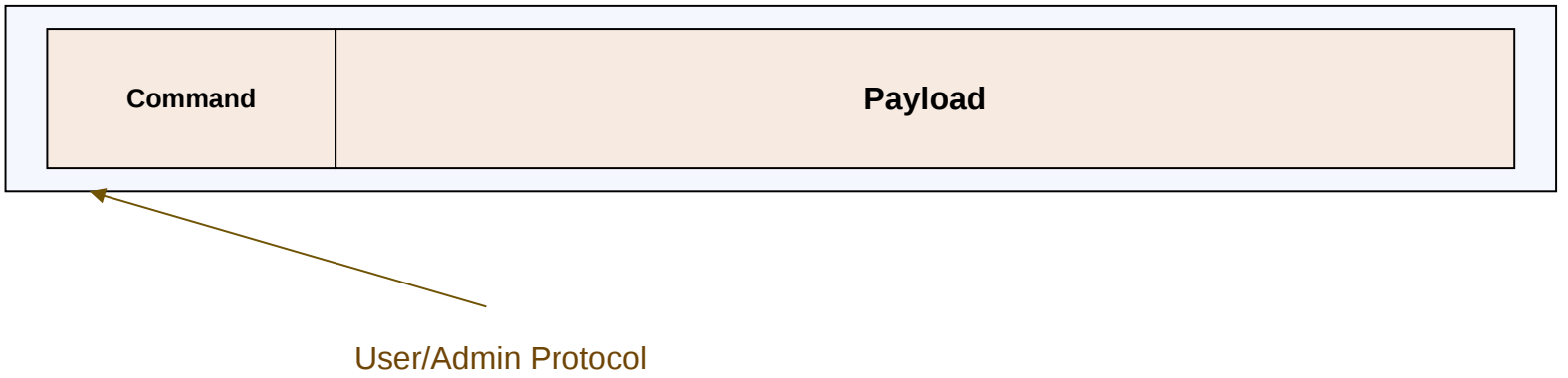
```
void UserButtonRelease(byte i) {
    unsetHandlerReceiveBuffer(i);
    unsetHandlerReceiveFunction(i);
}
```

Recepción de datos (user module protocol) :

```
void UserButtonReceived(byte* recBuffPtr, byte len, byte handler) {
    byte j;
    byte userButtonCounter = 0;
    switch (((BUTTON_DATA_PACKET*) recBuffPtr)->CMD) {
        case GET_VALUE:
            ((BUTTON_DATA_PACKET*) sendBufferUsrButton)->_byte[0] = ((BUTTON_DATA_PACKET*) recBuffPtr)->_byte[0];
            ((BUTTON_DATA_PACKET*) sendBufferUsrButton)->_byte[1] = getPortDescriptor(handler)->get_data_digital();
            userButtonCounter = 0x02;
            break;
    }
    USBGenWrite2(handler, userButtonCounter);
}/*end UserButtonReceived*/
```

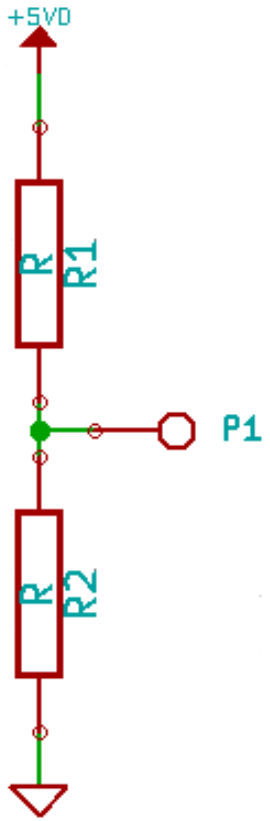
User Protocol

- Utilizada para que el usuario especifique las funcionalidades que el sensor/actuador expone.
- Command: Funcionalidad expuesta por el módulo
- Payload: Parámetros



Plug&Play

- Cada dispositivo dispone de una resistencia de identificación (R1).
- La placa posee una resistencia de valor conocido en cada puerto (R2)



- El valor en el punto P1 puede calcularse mediante la siguiente fórmula:

$$V_{\mathbf{out}} = \frac{R_2}{R_1 + R_2} \cdot V_{\mathbf{in}}$$

Plug&Play

- El firmware se encarga de llamar a la función de inicialización del módulo al momento de detectarlo
- Para agregar un nuevo sensor se debe agregar una entrada en la tabla `table_device_id_resistance[]`

```
rom const device_resistance table_device_id_resistance[MAX_DEVICES] = {
    { "port", R_PORT_MIN, R_PORT_MAX},
    { "light", R_LIGHT_MIN, R_LIGHT_MAX},
    { "button", R_BOTON_MIN, R_BOTON_MAX},
    { "grey", R_GREY_MIN, R_GREY_MAX},
    { "distanc", R_DIST_MIN, R_DIST_MAX},
    { "led", R_LED_MIN, R_LED_MAX}
};
```

Bobot

- Bobot[3] es un agente altamente portable y liviano que exporta la funcionalidad de los componentes USB4all presentes de una forma fácil de utilizar.
- Ofrece varios métodos de acceso:
  - Sockets
  - Web
- Bobot introduce el concepto de driver, en el cual se codifica el protocolo que utilizan los user modules

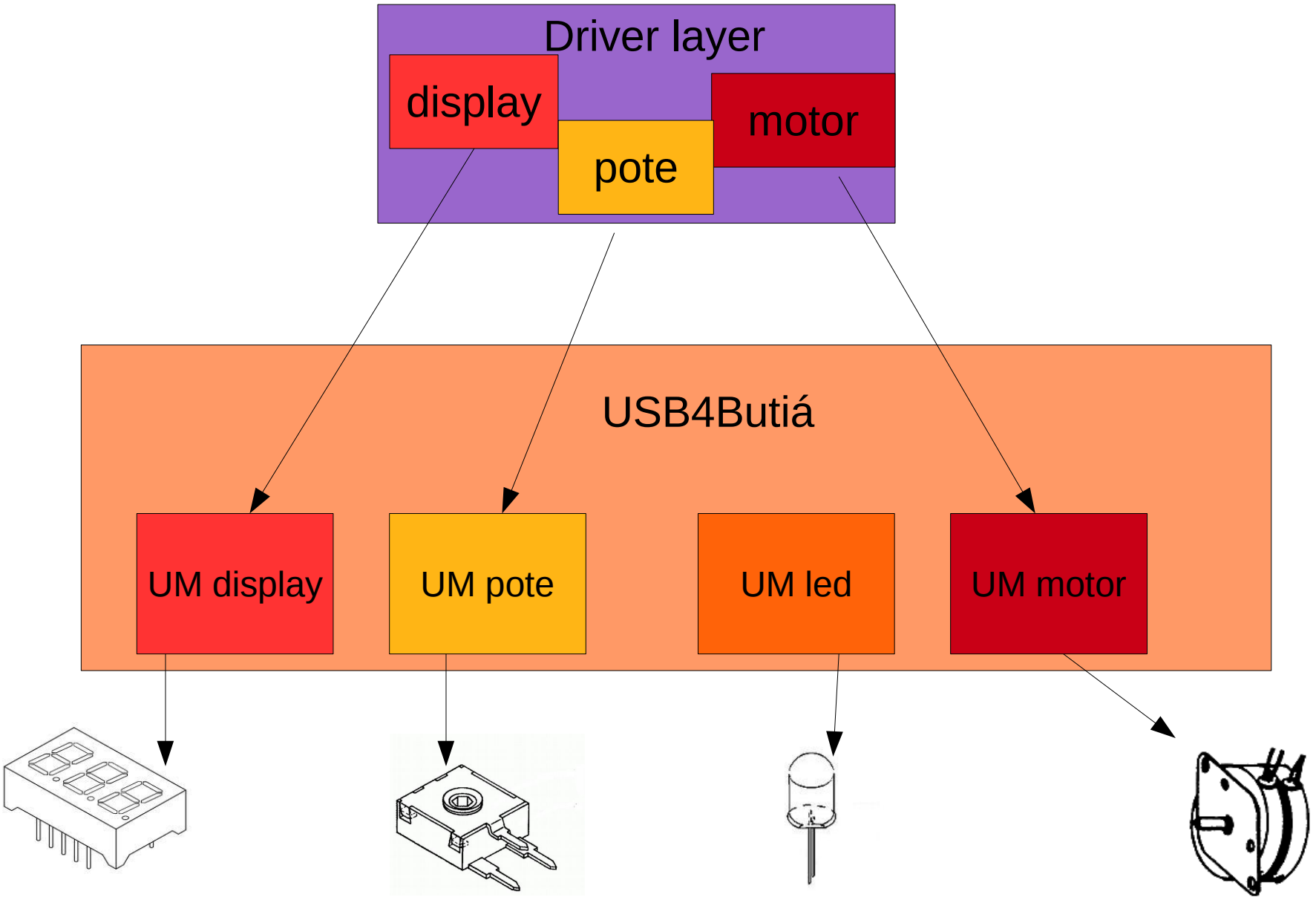
Drivers

- Componentes de software que interactúa directamente con los *user modules*
- Utiliza protocolo *user protocol*
- El código del driver es independiente de la tecnología de comunicación
  - Brindan información a los clientes acerca de las funcionalidades que implementan.
  - Nombre de la funcionalidad.
  - Nombre y tipo de parámetros que recibe.
  - Nombre y tipo del valor de retorno.

Drivers (implementación)

- Un driver es implementado mediante un conjunto de funciones escritas en Lua las cuales exportan las funcionalidades de los *user modules*
  - Agrupadas en una estructura de tabla en el archivo `nombreModulo.lua` almacenado en el directorio `bobot/drivers`
  - En la tabla donde se almacenan las funciones del driver, también se almacenan metadatos que especifican información sobre los valores de retorno y parámetros de las funciones

Drivers: ejemplo



Ejemplo: Driver `distance.lua`

```
local device = _G


local RD_VERSION=string.char(0x00)
local GET_VALUE=string.char(0x01)
local string_byte=string.byte
api={}
-- description: lets us know button's current status
-- input: empty
-- output: button's current status. Possible status: 1 pressed - 0 not pressed
api.getValue = {}
api.getValue.parameters = {} -- no input parameters
api.getValue.returns = {[1]={rname="state", rtype="int"}}
api.getValue.call = function ()
    device:send(GET_VALUE) -- operation code 1 = get button's status
    local sen_dig_response = device:read(2) -- 2 bytes to read (opcode, data)
    if not sen_dig_response or #sen_dig_response~=2 then return -1 end
        local raw_val = string_byte(sen_dig_response, 2) or 0 -- keep data
        return 1 - tonumber(raw_val)
end
```

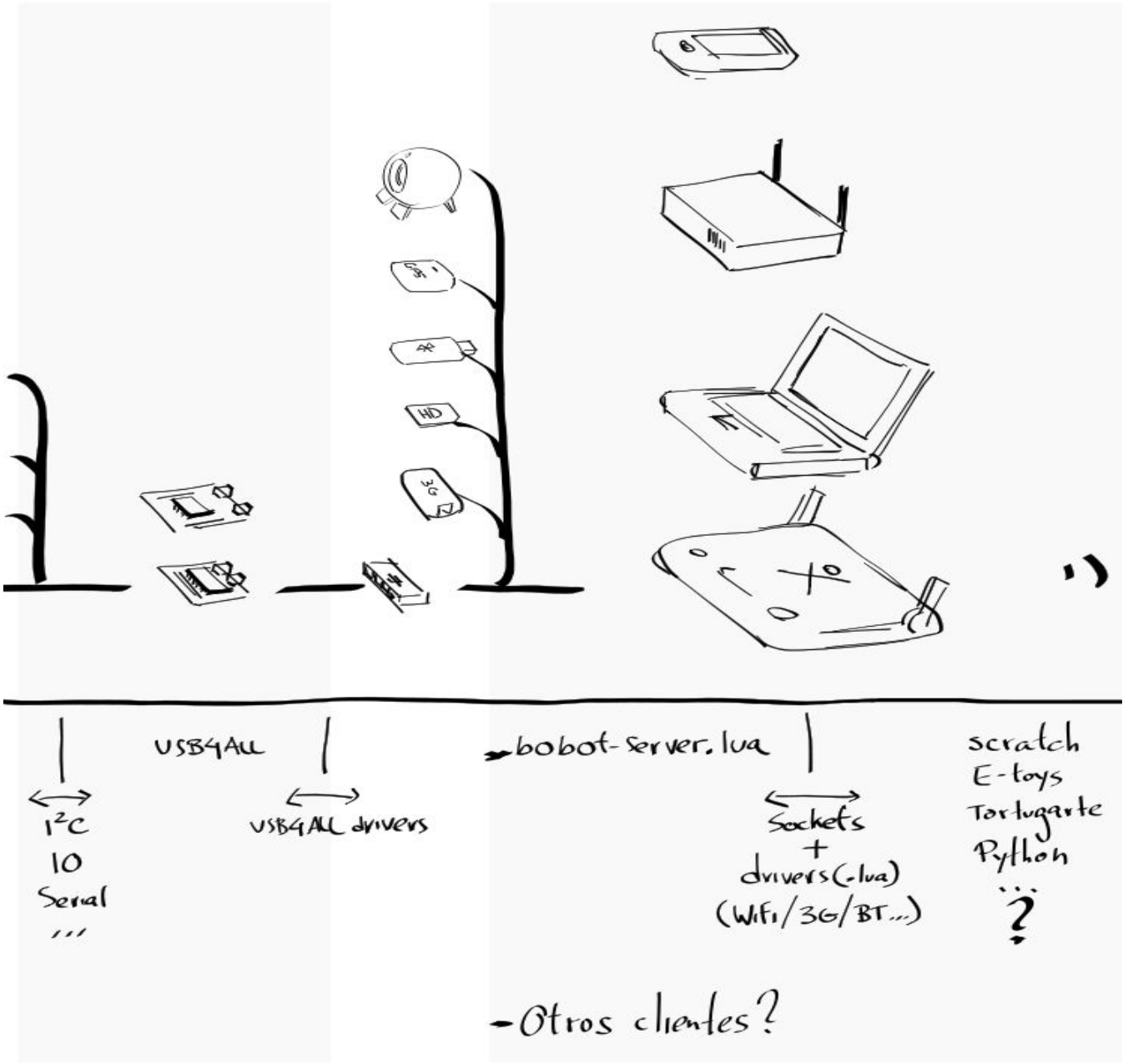
Bobot Server

- Brinda una interfaz de alto nivel para poder interactuar con los módulos (sensores/actuadores)
  - `LIST`
    - Lista los módulos detectados.
  - `DESCRIBE moduleName`
    - Devuelve una descripción del módulo.
  - `CALL moduleName operation param1, param2, ... , paramN`
    - Invoca la función indicada en el módulo dado. Los parámetros dependen de la función.
  - `CLOSEALL`
    - Cierra todos los módulos.
  - `OPEN moduleName`
    - Abre el módulo.
- Expone los servicios de los *user modules* en la red
- Facilita utilizar el robot desde diferentes lenguajes de programación

Bobot Server

- Protocolo orientado a mensajes sobre TCP (Protocolo de Control de Transmisión, utilizado en Internet) por defecto en el puerto 2009.
- Para utilizarlo en las aplicaciones de usuario deben abrir una conexión TCP e implementar dicho protocolo.
- Se puede interactuar directamente desde un terminal Telnet usando TCP.
- En el puerto 2010 se puede acceder a una versión que consume toda la información de los drivers y lo despliega en una web donde se puede interactuar con los sensores/actuadores conectados

Escalando ...


USB4ALL
bobot-server.lua
I²C
IO
Serial
...
USB4ALL drivers
Sockets
+
drivers(-lua)
(WiFi/3G/BT...)
scratch
E-toys
Tortugarte
Python
...
?
-Otros clientes?

Referencias

- **[1] USB4Butiá:** A truly free as in freedom input/output board **http://www.olpcnews.com/use_cases/technology/usb4butia_a_truly_free_as_in_freedom_input_output_board.html**
- **[2] USB4all:** Interfaz USB genérica para comunicación con dispositivos electrónicos, Aguirre, Fernández, Grossy http://www.fing.edu.uy/inco/grupos/mina/pGrado/pgusb/material.html
- **[3] Bobot:** http://www.fing.edu.uy/inco/proyectos/butia/mediawiki/index.php/Bobot