Klawiatura z Bluetooth

Realizacji interfejsu HID za pośrednictwem Raspberry PI

Bluetooth to powszechnie stosowany interfejs komunikacji bezprzewodowej na krótkie odległości. Niestety, jego praktyczna implementacja w projektach nastręcza bardzo wielu różnych problemów – czy to ze względu na konieczność przygotowywania odpowiednich profili BT, czy też z uwagi na ograniczoną dostępność dokumentacji i narzędzi. Zresztą korzystanie z komunikacji Bluetooth nie jest też takie łatwe, gdy ma się dostępne biblioteki i chce się jedynie oprogramować gotowe urządzenie. Ten ostatni problem można jednak obejść stosując profil HID – a więc interfejsu klawiatury i myszki. W artykule pokazano klawiaturę bezprzewodową, wykonaną w oparciu o Raspberry PI i tani modem BT na USB.

Zaprezentowany projekt może wydawać się bezsensowny. Do wykonania klawiatury z Bluetooth użyto zwykłej klawiatury USB, komputera Raspberry PI z kartą pamięci i zasilaczem oraz interfejs Bluetooth na USB. Koszt komponentów znacząco przewyższa cenę zakupu nawet całkiem dobrej klawiatury Bluetooth, a wykonany w ten sposób „produkt", o ile pozwala bezprzewodowo komunikować się np. ze smartfonem, to i tak w sumie musi być dołączony na stałe kablem do zasilania.

Projekt ten ma jednak bardzo dużą wartość dydaktyczną, gdyż zastosowany w nim mechanizm można łatwo zreplikować i wykorzystać do bezprzewodowego sterowania np. aplikacją mobilną przez miniaturowy komputer. Kluczowe jest tu użycie profilu HID, który natywnie obsługują praktycznie wszystkie urządzenia komputerowe z interfejsem Bluetooth. Co więcej, domyślnie jest on automatycznie interpretowany jako dane wprowadzane z klawiatury, co oznacza że Raspberry PI lub dowolny inny minikomputer może bezprzewodowo symulować wprowadzanie znaków na dowolnym urządzeniu mobilnym. Sekwencja przesyłanych znaków może reprezentować dowolne zdarzenia. Można w ten sposób przekazywać odczyty z czujników lub wykonać nietypowy interfejs człowiek-maszyna. Co więcej, by pozyskiwać odczyty nie trzeba nawet tworzyć aplikacji na urządzeniu mobilnym – wystarczy włączyć program notatnika i tam czekać na pojawiające się dane. Oczywiście, w praktyce byłoby to niewygodne, ale dla kreatywnego twórcy taki interfejs prawdopodobnie znajdzie zastosowanie, czy też przyda się na etapie prototypowania.

Problemy

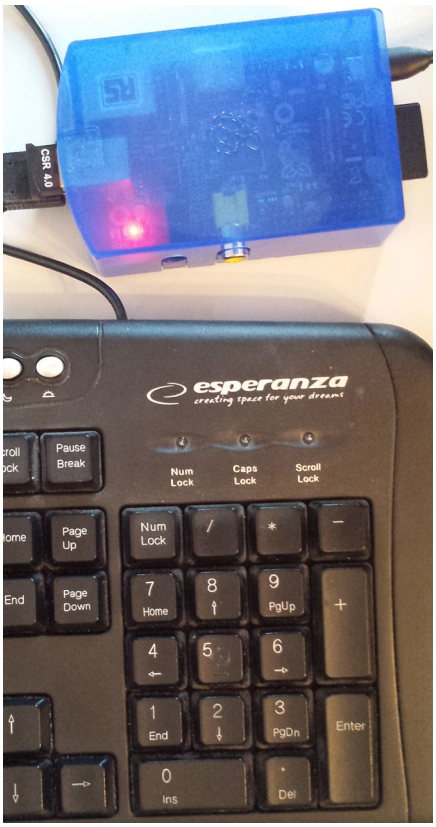
Wykonanie omówionego projektu mogłoby wydawać się całkiem łatwe – wystarczyłoby napisać program sczytujący odpowiednie dane i przesyłający je przez interfejs Bluetooth z użyciem gotowych bibliotek programowych. Problem jednak w tym, że przynajmniej pod kontrolą systemu Linux, wsparcie dla Bluetooth jest ograniczone.

Najbardziej popularnym zestawem bibliotek Bluetooth jest BlueZ, przy czym przez niektórych użytkowników jest on określany mianem „najskrzętniej pilnowanej tajemnicy Linuksa". Dotyczy to przede wszystkim dostępności dokumentacji, która nie tylko jest skąpa, co zawiera błędy. Sama biblioteka też dopracowana nie jest i nierzadko bywa, że gotowe przykłady z Internetu nie działają.

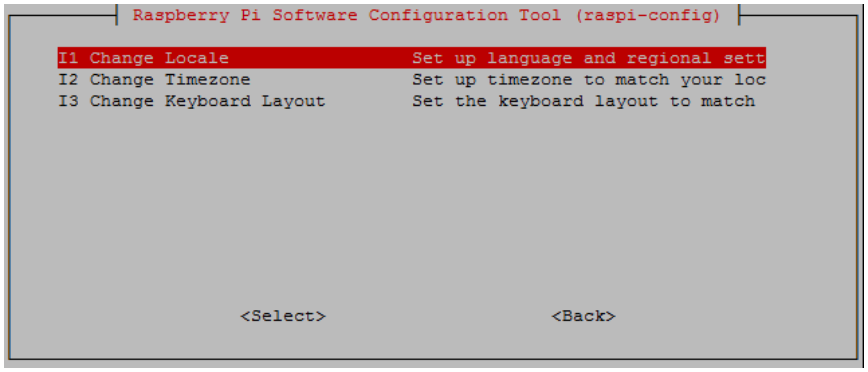
Na przestrzeni lat powstało wiele poradników pokazujących jak korzystać z Bluetootha pod Linuksem, a w tym także – jak używać BlueZ z poziomu języka Python. Niestety, kolejne wersje BlueZ sprawiły, że wiele z tych rad straciło na aktualności. Kluczowe jest rozróżnienie dwóch grup wersji BlueZ. Wraz z wprowadzeniem BlueZ 5.0 drastycznie zmodyfikowano dostępne API, usuwając wiele podstawowych funkcji. Zamiast tego zaimplementowano kilka innych, przy czym celem twórców BlueZ było wyeliminowanie zaszłości historycznych oraz dublujących się mechanizmów. Choć zamysł ten wydaje się bardzo szczytny, efektem takiego działania jest całkowity brak kompatybilności wstecznej BlueZ 5.0 i nowszych bibliotek. Co więcej, pomimo że najnowszą w tej chwili wersją BlueZ jest 5.37, wielu błędów wciąż nie udało się wyeliminować. W efekcie, ten sam interfejs USB Bluetooth na tym samym urządzeniu, z tym samym systemem operacyjnym może działać bezproblemowo z BlueZ starszym niż 5.0, podczas gdy aktualizacja narzędzi BlueZ zupełnie popsuje funkcjonowanie komunikacji. I dotyczy to nie tylko dostępnego API, które jest po prostu różne i niekompatybilne wstecz, ale także oficjalnych, skompilowanych na daną platformę narzędzi uruchamianych z linii poleceń.

Dodatkowe informacje
Dziękujemy firmie RS Components za dostarczenie Raspberry PI, w oparciu o które zrealizowano opisywany projekt.

Problem ten od niedawna dotyczy także użytkowników Raspberry PI. Wprowadzony we wrześniu Raspbian Jessie zawiera jądro Linuxa w wersji 4.1 i BlueZ 5.23. Natomiast oficjalnie dostępny na stronach Raspberry PI jeszcze do około połowy lutego, Raspbian Wheezy



Fotografia 1. Zmontowany, gotowy zestaw: klawiatura, Raspberry PI z kartą pamięci, zasilaczem i obudową oraz moduł Bluetooth na USB



Rysunek 2. Menu *raspi-config*, umożliwiające zmianę ustawień regionalnych

```
pi@raspberrypi ~ $ sudo hciconfig -a hci0
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10  SCO MTU: 64:8
        UP RUNNING PSCAN
        RX bytes:1483 acl:0 sco:0 events:73 errors:0
        TX bytes:1276 acl:0 sco:0 commands:73 errors:0
        Features: 0xff 0xff 0x8f 0xfe 0xdb 0xff 0x5b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT
        Name: 'raspberrypi-0'
        Class: 0x000100
        Service Classes: Unspecified
        Device Class: Computer, Uncategorized
        HCI Version: 4.0 (0x6)  Revision: 0x22bb
        LMP Version: 4.0 (0x6)  Subversion: 0x22bb
        Manufacturer: Cambridge Silicon Radio (10)
```

Rysunek 3. Parametry interfejsu hci0 po konfiguracji, ale w trakcie gdy stworzony program jest wyłączony

korzysta ze starszego jądra i z BlueZ 4.99. Starsze aplikacje, napisane pod kątem BlueZ 4.x nie będą więc poprawnie działały w Raspbianie Jessie. Starszą wersję Raspbiana, choć nie widać jej na głównych stronach pobierania Raspberry PI, można zdobyć z adresu: *http://goo.gl/fnRaBI*. Inne archiwalne wersje Raspbiana można znaleźć tu *http://goo.gl/xjgocJ*.

Podłączenie elementów

W celu stworzenia opisanej klawiatury bezprzewodowej do Raspberry PI podłączamy klawiaturę USB oraz kartę Bluetooth na USB. W naszym przypadku korzystamy z modelu opartego o układ CSR, zgodnego z Bluetooth 4.0, choć nie będziemy potrzebowali funkcji Bluetooth Smart. Raspberry PI podłączamy do prądu z odpowiednio mocnym zasilaczem. Z pomiarów wynika, że pobór prądu w trakcie pracy takiego zestawu nie przekracza 0,5 A, ale lepiej zaopatrzyć się w mocniejszy zasilacz, ponieważ zastosowany przez nas miernik mógł nie wykrywać krótkotrwałych pików w poborze mocy.

Załadowany na kartę SD obraz systemu Raspbian Wheezy z BlueZ 4.99 pozwala na uruchomienie komputera. Tu dostępne są dwie możliwości: Raspberry PI można podpiąć do sieci i korzystać przez SSH lub podłączyć do wyświetlacza i korzystać w normalny sposób. Ta druga opcja jest koniecznością, gdy posiada się tylko jedną klawiaturę. Pierwsza natomiast może być wygodniejsza, ale trzeba też zwrócić uwagę na jedną rzecz. W trakcie pracy programu wszystkie treści wpisywane z klawiatury są nie tylko przetwarzane przez program, uruchomiony – czy to zdalnie, czy w ramach pierwszej otwartej powłoki, ale też interpretowane przez podstawową instancje powłoki. Inaczej mówiąc – wszystkie wpisywane z klawiatury znaki trafiają do interpretera poleceń. Oznacza to, że można w ten sposób „przez przypadek" wyłączyć Raspberry PI lub nawet usunąć cały system plików. Na szczęście, jeśli profil Bluetooth HID jest wykorzystywany do przesyłania innego rodzaju danych niż pochodzące z prawdziwej klawiatury, tj. np. zdobywane z czujników, problem ten przestaje mieć znaczenie.

Aby móc w pełni skorzystać z Bluetootha z poziomu języka Python oraz by móc reagować na zdarzenia, trzeba zainstalować dodatkowe moduły. Te, by się zmieściły, potrzebują większej przestrzeni niż domyślnie zarezerwowana na system plików. Dlatego konieczne staje się jej powiększenie z użyciem menu **raspi-config**, wywołanego z uprawnieniami roota (**sudo**). W omawianej wersji systemu pierwsza pozycja na liście odpowiada za rozszerzanie systemu plików do maksymalnych rozmiarów. Od razu warto też zajrzeć do ustawień narodowych (pozycja 4 w menu **raspi-config**) i dobrać tam ustawienia języka, strefę czasową oraz układ klawiatury. Znaczenie ma szczególnie ostatnia z tych czynności, gdyż w przeciwnym wypadku klawisze przesyłane z tak powstałej klawiatury nie będą odpowiadały tym rzeczywiście wciskanym – systemowi będzie się wydawało, że podłączona jest klawiatura brytyjska, różnią się układem od tej stosowanej w Polsce. Po wprowadzeniu tych wszystkich ustawień należy zrestartować system.

Do wykonania projektu będziemy potrzebowali następujących programów: *bluez*, *bluez-tools*, *bluez-firmware*, *python-bluez*, *python-dev*, *python-gobject*, *python-pip*. Instalujemy je za pomocą polecenia **apt-get install**, po uprzednim uruchomieniu **apt-get update** (w obu wypadkach z uprawnieniami roota). Ostatni z programów to menedżer pakietów Pythona. Za jego pomocą da się zainstalować potrzebny moduł **evdev**, który służy do przekazywania zdarzeń jądra systemu bezpośrednio do Pythona (i w drugą stronę). U nas potrzebny jest do wykrywania zdarzeń z klawiatury i reagowania na nie. Evdev trzeba zainstalować za pomocą programy pip (polecenie **sudo pip install evdev**), gdyż w ten sposób moduł ten jest kompilowany w trakcie instalacji, pomimo że nie ma go skompilowanego, dostępnego dla Raspberry PI.

Może się też okazać, że po zainstalowaniu powyżej opisanych programów, domyślnie uruchomione zostaną pewne usługi Bluetoothowe. By je wyłączyć należy edytować plik **/etc/bluetooth/main.conf** i upewnić się, że istnieje w nim linijka DisablePlugins, której przypisane są odpowiednie tryby oraz że nie jest ona wykomentowana. Powinna wyglądać następująco: *DisablePlugins = network,input,audio,pnat,sap,serial*. Po zapisaniu pliku konfiguracyjnego należy zrestartować demona interfejsu Bluetooth (polecenie **sudo /etc/init.d/bluetooth restart**).

Obsługa klawiatury USB

Cały program można podzielić na dwie główne części. Jedna obsługuje klawiaturę USB sczytując wciskane klawisze i przygotowuje dane do wysłania dowolnym interfejsem. Druga część dotyczy interfejsu Bluetooth – pozwala na jego konfigurację, nawiązywanie połączenia oraz na przesyłanie dowolnych danych, jakie zostaną przygotowane. Python pozwala programować obiektowo i takie właśnie podejście wydaje się tu najlepsze – szczególnie że dzięki temu za pomocą przygotowanych funkcji Bluetoothowych można będzie następnie przesyłać dowolne inne dane – np. pochodzące z czujników.

Obsługę klawiatury można zacząć od jej inicjalizacji. Można ją znaleźć za pomocą funkcji *InputDevice(„/dev/input/event0”)*. Pod taką właśnie ścieżką powinna być wykrywana, o ile tylko podłączono jedną klawiaturę USB do komputera. Trzeba też przygotować funkcję nasłuchującą zdarzeń z klawiatury. Powinna ona iterować zdarzenia odbierane za pomocą funkcji *device.read_loop()*, gdzie **device** to wartość zwrócona przez wcześniej wskazaną funkcję **InputDevice()**, a więc wskazująca na klawiaturę właśnie. Następnie trzeba sprawdzić, czy znalezione zdarzenie to naciśnięcie lub puszczenie klawisza. Można to zrobić wpisując `if event.type == ecodes.EV_KEY and event.value < 2:`, gdzie **event** to aktualnie przeglądane zdarzenie, a **ecodes.EV_KEY** to stała. Jeśli zdarzenie jest tym poszukiwanym, należy kolejno przetworzyć zdarzenie na odpowiedni format, opisujący naciśniętą konfigurację klawiszy, a następnie można już wysłać przygotowaną w ten sposób wartość za pomocą interfejsu Bluetooth.

Przetworzenie informacji o naciśniętych klawiszach do formatu, jakiego używają klawiatury Bluetooth nie jest trywialne. Trzeba pamiętać, że klawiatura może mieć naciśniętych jednocześnie kilka klawiszy, a w szczególności, że Shift, Alt i Ctrl zmieniają znaczenie pozostałych naciśniętych klawiszy. W końcu i sam moduł obsługi zdarzeń nie zwraca kodów klawiatury w sposób numeryczny, tylko za pomocą stałych tekstowych. Ich lista, wraz z wartościami numerycznymi, zgodnymi z profilem Bluetooth HID została zebrana w **tabeli 1**. Uzupełnieniem jest **tabela 2**, w której znajdują się zdarzenia specjalne, opisujące naciśnięcie klawiszy Windows, ALT, Shift i Control, kolejno po prawej i lewej stronie. Ich kody HID pokrywają się z niektórymi kodami zwykłych klawiszy, ale są przesyłane w innym miejscu pakietu danych transferowanych przez Bluetooth, dlatego nie stanowi to problemu. W praktyce są to właśnie pozycje bitów w bajcie, informującym o wciśniętej konfiguracji klawiszy specjalnych.

Obsługa całej klawiatury wymaga sprawdzenia, z którym zdarzeniem związane jest dane zdarzenie oraz czy klawisz został właśnie naciśnięty, czy puszczony. Ponieważ przeglądane zdarzenia nie zawierają informacji o klawiszach, które wciąż są wciśnięte, albo wciąż puszczone, a jedynie tych, których stan się zmienił, aby mieć poprawny obraz tego, co aktualnie jest wciśnięte, trzeba samodzielnie zapamiętywać stan klawiszy.

Sprawdzenie, czy zdarzenia dotyczą klawiszy specjalnych jest prostsze – jest 8 klawiszy, każdy z nich ma swój własny bit w bajcie reprezentującym ich stan. Jeśli kod zdarzenia odpowiada któremuś z kodów klawiszy specjalnych należy w pamiętanym bajcie stanu odwrócić bit na pozycji odpowiadającej temu klawiszowi. Kod użytego klawisza odczytywany jest z tablicy w następujący sposób

Tabela 1. Zdarzenia zgłaszane przez moduł evdev oraz odpowiadające im kody klawiszy profilu Bluetooth HID

Zdarzenie	Kod HID	Zdarzenie	Kod HID	Zdarzenie	Kod HID
KEY_RESERVED	0	KEY_RIGHTSHIFT	229	KEY_INSERT	73
KEY_ESC	41	KEY_KPASTERISK	85	KEY_DELETE	76
KEY_1	30	KEY_LEFTALT	226	KEY_MUTE	239
KEY_2	31	KEY_SPACE	44	KEY_VOLUME-DOWN	238
KEY_3	32	KEY_CAPSLOCK	57	KEY_VOLU-MEUP	237
KEY_4	33	KEY_F1	58	KEY_POWER	102
KEY_5	34	61- KEY_F2	59	KEY_KPEQUAL	103
KEY_6	35	62- KEY_F3	60	KEY_PAUSE	72
KEY_7	36	KEY_F4	61	KEY_KPCOMMA	133
KEY_8	37	KEY_F5	62	KEY_HANGEUL	144
KEY_9	38	KEY_F6	63	KEY_HANJA	145
KEY_0	39	KEY_F7	64	KEY_YEN	137
KEY_MINUS	45	KEY_F8	65	KEY_LEFTMETA	227
KEY_EQUAL	46	KEY_F9	66	KEY_RIGHT-META	231
KEY_BACKSPACE	42	KEY_F10	67	KEY_COMPOSE	101
KEY_TAB	43	KEY_NUMLOCK	83	KEY_STOP	243
KEY_Q	20	KEY_SCROL-LLOCK	71	KEY_AGAIN	121
KEY_W	26	KEY_KP7	95	KEY_PROPS	118
KEY_E	8	KEY_KP8	96	KEY_UNDO	122
KEY_R	21	KEY_KP9	97	KEY_FRONT	119
KEY_T	23	KEY_KPMINUS	86	KEY_COPY	124
KEY_Y	28	KEY_KP4	92	KEY_OPEN	116
KEY_U	24	KEY_KP5	93	KEY_PASTE	125
KEY_I	12	KEY_KP6	94	KEY_FIND	244
KEY_O	18	KEY_KPPLUS	87	KEY_CUT	123
KEY_P	19	KEY_KP1	89	KEY_HELP	117
KEY_LEFTBRACE	47	KEY_KP2	90	KEY_CALC	251
KEY_RIGHT-BRACE	48	KEY_KP3	91	KEY_SLEEP	248

Tabela 1. c.d.

Zdarzenie	Kod HID	Zdarzenie	Kod HID	Zdarzenie	Kod HID
KEY_ENTER	40	KEY_KP0	98	KEY_WWW	240
KEY_LEFTCTRL	224	KEY_KPDOT	99	KEY_COFFEE	249
KEY_A	4	KEY_ZENKA-KUHANKAKU	148	KEY_BACK	241
KEY_S	22	KEY_102ND	100	KEY_FORWARD	242
KEY_D	7	KEY_F11	68	KEY_EJECTCD	236
KEY_F	9	KEY_F12	69	KEY_NEXT-SONG	235
KEY_G	10	KEY_RO	135	KEY_PLAY-PAUSE	232
KEY_H	11	KEY_KATAKANA	146	KEY_PREVIO-USSONG	234
KEY_J	13	KEY_HIRAGANA	147	KEY_STOPCD	233
KEY_K	14	KEY_HENKAN	138	KEY_REFRESH	250
KEY_L	15	KEY_KATAKANA-HIRAGANA	136	KEY_EDIT	247
KEY_SEMICO-LON	51	KEY_MUHENKAN	139	148 - KEY_SCROLLUP	245
KEY_APOSTRO-PHE	52	KEY_KPJPCOMMA	140	KEY_SCROLL-DOWN	246
KEY_GRAVE	53	KEY_KPENTER	88	KEY_F13	104
KEY_LEFTSHIFT	225	KEY_RIGHTCTRL	228	KEY_F14	105
KEY_BACKSLA-SH	50	KEY_KPSLASH	84	KEY_F15	106
KEY_Z	29	KEY_SYSRQ	70	KEY_F16	107
KEY_X	27	KEY_RIGHTALT	230	KEY_F17	108
KEY_C	6	KEY_HOME	74	KEY_F18	109
KEY_V	25	KEY_UP	82	KEY_F19	110
KEY_B	5	KEY_PAGEUP	75	KEY_F20	111
KEY_N	17	KEY_LEFT	80	KEY_F21	112
KEY_M	16	105 - KEY_RI-GHT	79	KEY_F22	113
KEY_COMMA	54	KEY_END	77	KEY_F23	114
KEY_DOT	55	KEY_DOWN	81	161 - KEY_F24	115
KEY_SLASH	56	KEY_PAGEDOWN	78		

Tabela 2. Zdarzenia związane z naciśnięciem klawiszy specjalnych oraz odpowiadające im pozycje w pakiecie danych przesyłanych zgodnie z profilem Bluetooth HID

Zdarzenie	Pozycja
KEY_RIGHTMETA	0
KEY_RIGHTALT	1
KEY_RIGHTSHIFT	2
KEY_RIGHTCTRL	3
KEY_LEFTMETA	4
KEY_LEFTALT	5
KEY_LEFTSHIFT	6
KEY_LEFTCTRL	7

ecodes.KEY[event.code], a pozycję można odnaleźć tworząc kolejną tablicę, w oparciu o tabelę 2.

Sprawdzenie, czy zdarzenie dotyczy pozostałych klawiszy jest o tyle trudniejsze, że nie jesteśmy (a właściwie to nie powinniśmy) zapamiętywać zbyt dużej liczby jednocześnie naciśniętych klawiszy. Profil Bluetooth HID obejmuje jedynie (lub aż!) 6 pozycji na których można zamieścić kody jednocześnie wciśniętych zwykłych klawiszy. Dlatego dla każdego zdarzenia należy najpierw sprawdzić, jaki kod odpowiada danemu klawiszowi, następnie sprawdzić, czy kod ten został już zapamiętany na jednej z 6 dostępnych pozycji. Jeśli tak, to trzeba dowiedzieć się, czy zdarzenie dotyczy naciśnięcia (event.value==1) czy puszczenia klawisza (event.value==0). Jest to o tyle istotne, że jeśli naciśniętych zostanie jednocześnie więcej niż 6 klawiszy, nie wszystkie zostaną zapamiętane i w momencie, gdy jeden z zapamiętanych zostanie puszczony, a następnie puszczony zostanie jeden który nie został wcześniej zapamiętany, bez sprawdzenia, że właśnie go puszczono system domyślnie by błędnie przyjął, że nastąpiło naciśnięcie, a nie puszczenie.

Oczywiście, jeśli zdarzenie dotyczy puszczenia klawisza,, który wcześniej był zapamiętany – należy odnaleźć pozycję, na której został zapisany i ją wyzerować. Jeśli zdarzenie dotyczy naciśnięcia klawisza, trzeba znaleźć pierwszą wolną (wyzerowaną) pozycję i tam go zapisać, po czym można przejść do ewentualnej wysyłki przez Bluetooth zapisanego stanu klawiszy.

Format danych Bluetooth HID Keyboard

Aby urządzenie Bluetooth było rozpoznawane jako klawiatura zgodna z profilem Bluetooth HID, trzeba je odpowiednio skonfigurować, a transmitowane dane muszą mieć dobrze rozpoznawany format. Konfiguracja opiera się o przygotowanie pliku w Service Discovery Protocol (SDP) w formacie xml, w którym zawarte są informacje o danym urządzeniu. Do tworzenia tego pliku służy linuksowe narzędzie **sdptool**, znajdujące się w pakiecie BlueZ. Można też posłużyć się gotowym plikiem – jego samodzielne stworzenie nie jest łatwe. Plik ten zawiera unikalny identyfikator usługi danego urządzenia Bluetooth jako węzeł umieszczony wewnątrz węzła **attribute** o identyfikatorze **0x0001**. UUID dla klawiatury HID ma wartość **0x1124**.W pliku SDP znajduje się też m.in. nazwa urządzenia, jaka będzie widziana poprzez urządzenia przeszukujące otoczenie w poszukiwaniu sprzętu Bluetooth.

W omawianym przypadku, plik SDP narzuca zastosowanie konkretnego formatu danych przesyłanych z urządzenia. W przypadku wystąpienia dowolnego zdarzenia związanego z naciśnięciem lub puszczeniem klawisza na klawiaturze, program wysyła 10-bajtowy pakiet danych. Trzy bajty są stałe. Pierwszy to zawsze 0xA1, informujący że przesyłany pakiet dotyczy wprowadzonych przez użytkownika danych. Drugi bajt to 0x01 – informujący, że dane mają być interpretowane, jakby były wprowadzone z klawiatury. Trzeci bajt stały nie jest – jest on tworzony w oparciu o klawisze specjalne, w których kolejne bity odpowiadają stanom klawiszy, zgodnie z tabelą 2. Czwarty bajt jest stały – wynosi 0x00. Pozostałe 6 bajtów odpowiada natomiast kodom aktualnie wciśniętych klawiszy, zgodnie z tabelą 1. Cały pakiet został zaprezentowany w **tabeli 3**. Co istotne, pakiet jest przesyłany w postaci ciągu znaków, których kody ASCII odpowiadają kodom HID klawiszy. Oznacza to, że przykładowo – klawisz F8, którego kod HID to 65 jest przesyłany jako duża litera A umieszczona na pozycji z zakresu 5…10.

Inicjalizacja Bluetooth

Część związaną z obsługą samego interfejsu Bluetooth warto zawrzeć w oddzielnej klasie. Ta wymaga zdefiniowania kilku stałych, inicjalizacji interfejsu, stworzenia funkcji nasłuchującej na połączenie zewnętrznego urządzenia BT oraz wysyłającej otrzymany pakiet danych.

Inicjalizację można przeprowadzić korzystając z zewnętrznych programów, zainstalowanych w ramach BlueZ. Jednym z nich jest **hciconfig**. Polecenie to służy do ustawiania parametrów interfejsu Bluetooth, nieco podobnie jak ustawia się np. interfejsy sieciowe. Kolejne interfejsy Bluetooth mają swoje nazwy. Pierwszy to zawsze **hci0**. Należy mu przypisać klasę reprezentującą klawiaturę, nazwę oraz umożliwić wykrywanie przez inne urządzenia. Służą do tego kolejno polecenia:

```
hciconfig hci0 class 0x002540
hciconfig hci0 name Raspberry\ Pi
hciconfig hci0 piscan
```

W skrypcie Pythona można je wywołać korzystając z funkcji os.system(polecenie), przy czym wymagają one uprawnień roota, więc i sam skrypt Pythona musi być uruchomiony z takimi uprawnieniami.

Obsługa profilu HID wymaga posługiwania się dwoma gniazdami (sockets) – jednym kontrolnym i jednym do przerwań. Pierwsze służy do nawiązywania połączenia, a drugie do przekazywania pakietów danych, takich jak np. wcześniej przygotowane 10 bajtów stanu naciśniętych klawiszy. Standardowo gniazdu kontrolnemu przypisuje się port 17, a gniazdu przerwań – 19.

Do przesyłania danych wykorzystywany będzie protokół L2CAP – Logical Link Control and Adaptation Protocol. Mechanizm ten przekazuje, dzieli i łączy pakiety, multipleksuje dane na potrzeby protokołów wyższego poziomu oraz zapewnia obsługę QoS. Funkcja BluetoothSocket pozwala na zdefiniowanie gniazd komunikacyjnych, zgodnie z wcześniej określonymi potrzebami, z użyciem protokołu L2CAP:

```
scontrol = BluetoothSocket(L2CAP)
sinterrupt =
BluetoothSocket(L2CAP).
```

Natomiast przypisanie do nich portów odbywa się z użyciem funkcji bind wywoływanej w następujący sposób:

```
scontrol.bind(„”, 17)
sinterrupt.bind(„”,19).
```

Moduł Pythona do obsługi Bluetootha, jak i cała biblioteka BlueZ, nie jest doskonały – nie obejmuje kilku potrzebnych funkcji, dlatego konieczne staje się użycie magistrali D-Bus – prostego systemu komunikacji międzyprocesorowej. System ten wymaga rejestracji usług, dostępnych dla innych procesów, z których następnie można dosyć swobodnie korzystać. W Pythonie do obsługi magistrali D-Bus wykorzystywana jest biblioteka dbus. Jednocześnie program BlueZ, zainstalowany w systemie, automatycznie rejestruje w ramach magistrali D-Bus potrzebne usługi. Dzięki nim można skonfigurować SDP z użyciem pliku w formacie XML, a więc rozgłaszać otoczeniu realizowane przez dane urządzenie funkcje. Trzeba jednak zaznaczyć, że zarejestrowane usługi są zupełnie inne w BlueZ 5.0 i nowszych, niż w starszych wersjach.

Dostęp do magistrali D-Bus uzyskuje się poprzez funkcję dbus.SystemBus(). Następnie trzeba sięgnąć

```
pi@raspberrypi ~ $ sudo hciconfig -a hci0
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:2705 acl:11 sco:0 events:110 errors:0
        TX bytes:2309 acl:10 sco:0 commands:88 errors:0
        Features: 0xff 0xff 0x8f 0xfe 0xdb 0xff 0x5b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT
        Name: 'Raspberry Pi'
        Class: 0x002540
        Service Classes: Unspecified
        Device Class: Peripheral, Keyboard
        HCI Version: 4.0 (0x6)  Revision: 0x22bb
        LMP Version: 4.0 (0x6)  Subversion: 0x22bb
        Manufacturer: Cambridge Silicon Radio (10)
```

Rysunek 4. Parametry interfejsu hci0 po konfiguracji i uruchomieniu programu. Warto zwrócić uwagę na zmianę klasy na 0x002540 i jej opisu na „Peripheral, Keyboard”

Tabela 3. Przesyłany pakiet danych i ich znaczenie

Pozycja (bajt)	1	2	3								4	5	6	7	8	9	10
Wartość	0xA1	0x01	X	X	X	X	X	X	X	X	0x00	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX
Znaczenie	Informacja o wciśniętych klawiszach	Dane wprowadzane przez klawiaturę	Lewy CTRL	Lewy SHIFT	Lewy ALT	Lewy Win	Prawy CTRL	Prawy SHIFT	PRAWY ALT	PRAWY WIN	Stały, zarezerwowany	Kody odpowiadające wciśniętym zwykłym klawiszom					

po tzw. interfejs, wybrać adapter Bluetooth oraz znaleźć usługę. Służą temu polecenia:

```
bus = dbus.SystemBus()
manager = dbus.Interface(-
bus.get_object("org.bluez",
"/"), "org.bluez.Manager")
adapter_path = man-
ager.DefaultAdapter()
service = dbus.Interface(bus.
get_object("org.bluez", adapter_
path), "org.bluez.Service")
```

Nazwy **org.bluez**, **org.bluez.Manager** i **org.bluez.Service** występują w BlueZ 4.99, ale nie ma ich już w BlueZ 5.0 – w nowszej wersji zupełnie zmieniono miejsca rejestracji usług magistrali D-Bus.

Załadowanie pliku SDP w formacie XML jako rekord opisu usługi odbywa się poleceniem `service.AddRecord(read_relative_file(„sdp_file.xml"))`. Można też po prostu wcześniej wczytać plik, a wprowadzić do rekordu opisu usługi dopiero, w momencie gdy urządzenie zaczyna nasłuchiwać w oczekiwaniu na połączenie:

```
fsdp=open(sys.path[0] + „/
sdp_file.xml","r")
service_record = fh.read()
fh.close()
```

Nasłuchiwanie i przyjmowanie połączenia

Gdy interfejs Bluetooth jest już skonfigurowany, gniazda są zainicjalizowane wraz z przypisanymi portami, usługa BlueZ w magistrali D-Bus odnaleziona, a plik SDP wczytany, można przejść do rozgłaszania możliwości urządzenia Bluetooth i oczekiwania na połączenie. Jeśli wcześniej plik SDP został tylko wczytany, ale nie załadowany jako rekord usługi Bluetooth przez D-Bus, należy to zrobić, np. korzystając z funkcji `service_handle = service.AddRecord(service_record)`. Można natychmiast zacząć nasłuchiwanie, najlepiej ograniczając liczbę jednoczesnych połączeń do 1. Nasłuchiwanie powinno być prowadzone jednocześnie na obu gniazdach/portach:

```
scontrol.listen(1)
sinterrupt.listen(1)
```

Urządzenie będzie można bez problemu wykryć w otoczeniu, dzięki wcześniej wydanemu poleceniu **hciconfig hci0 piscan**. Gdy tylko połączenie nadejdzie, trzeba je zaakceptować. Służą do tego funkcje `ccontrol, cinfo = scontrol.accept()` oraz `cinterrupt, cinfo = sinterrupt.accept()`. Po każdej z tych dwóch linijek, w tablicy **cinfo** na pozycji 0 znajduje się adres MAC urządzenia, które się podłączyło. Trzeba przy tym zaznaczyć, że formalnie jest to host, a Raspberry PI z podłączoną klawiaturą USB i interfejsem Bluetooth pracuje jako urządzenie podporządkowane.

Przesyłanie pakietów danych

Jak wspomniano, stan naciśniętych klawiszy na klawiaturze jest zapisywany w postaci 10 bajtów, które następnie tłumaczone są na znaki ASCII, w oparciu o te właśnie bajty. Gotowy ciąg znaków przesyłany jest przez gniazdo przerwań (port 19) poleceniem `cinterrupt.send(ciag_10_znakow)`.

Kolejność wykonywania

Aby całość działała poprawnie trzeba kolejno uruchomić funkcje odpowiadające za:

- konfigurowanie i inicjalizację interfejsu Bluetooth oraz załadowanie SDP,
- nasłuchiwanie i nawiązanie połączenia z hostem,
- wykrycie klawiatury,
- oczekiwanie na zdarzenia związane z klawiaturą i w razie potrzeby wysyłające pakiet danych przez interfejs Bluetooth.

Jak łatwo zauważyć, jeśli wykrywanie klawiatury zastąpić wykrywaniem wybranych sensorów, a funkcje oczekiwania na zdarzenia związane z klawiaturą zamienić na funkcje oczekujące na zdarzenia związane z czujnikami, lub regularnie, co określony czas, odpytujące czujniki, można by było przesyłać przez Bluetooth dowolne inne dane, które z poziomu hosta byłyby widziane tak, jakby je wprowadzać z klawiatury. Oczywiście wymagałoby to stosownego przetworzenia danych tak, by np. odczyt temperatury skutkował przesłaniem np. szeregu 10-bajtowych pakietów, w których kolejno będą pojawiać się cyfry odpowiadające kolejnym cyfrom zmierzonej wartości oraz znaki jednostek.

Parowanie urządzeń

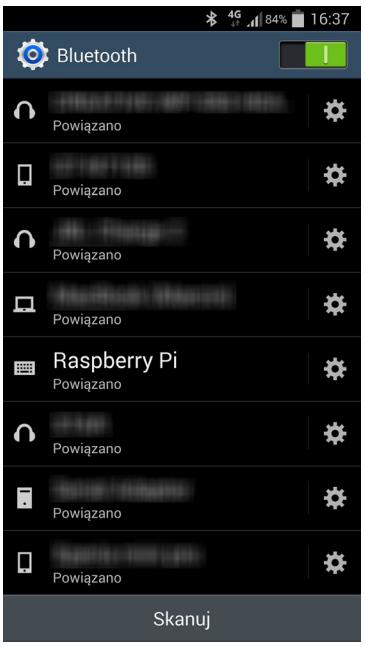
Bluetooth wymaga jeszcze, by urządzenia się ze sobą komunikujące były sparowane – dzięki temu nie ma wątpliwości, że dane połączenie nie jest niepowołane. W omawianym przypadku parowanie można przeprowadzić na kilka sposobów. Potrzebna jest znajomość adresu MAC urządzenia, z którym Raspberry PI będzie się łączyło. Adres ten należy podać w komendzie **bluez-simple-agent hci0 ADRES_MAC_URZADZENIA.**

Komenda musi być uruchomiona z prawami roota, po czym należy podać numer PIN, który zostanie następnie wpisany na urządzeniu podłączającym się do Raspberry PI. Adres MAC można zdobyć m.in. uruchamiając napisany na potrzeby tego projektu program, gdyż próba połączenia nawet niesparowanego urządzenia spowoduje zapisanie adresu do tablicy **cinfo**. Ręczna obsługa komend Bluetooth z poziomu linii poleceń jest znacznie łatwiejsza i wygodniejsza w BlueZ 5, choć nie w pełni działa.

Podsumowanie

Program realizujący opisany w artykule algorytm został napisany przez programistów z serwisu LinuxUser i można go pobrać z adresu *http://goo.gl/JapvFS*. Adaptacja kodu do nowszej wersji BlueZ nie jest łatwa, ale sama komunikacja za pomocą Bluetooth w Raspberry PI na pewno w niedługim czasie będzie częściej wykorzystywana. Wynika to z faktu, że najnowsze Raspberry PI 3 jest wyposażone we wbudowany układ Bluetooth, co zachęca do jego stosowania. Niemniej w przypadku posiadania starszego RPI opisany przykład powinien działać z dowolnym modułem BT na USB.

Marcin Karbowniczek, EP



Rysunek 5. Raspberry PI widziane przez telefon z Androidem, jako klawiatura Bluetooth