Google Assistant: Custom Actions

Microprocessor Systems and Applications

GITHUB.COM/CLAUDIUGROZA/MSA

April 2018

Contents

1. Google Assistant

1.1 Problem statement

This document aims to exemplify how Google Assistant SDK can be used to control the brightness of an LED. We'll learn how to define custom actions that will trigger different behavior on the Raspberry Pi board.

1.2 Prerequisites

In order to start the actual implementation, the following list of requirements must be satisfied before moving on to the next sections:

1. Configure audio input and sound output [Aud]
2. Create and configure Google Action developer project [Deva]
3. Register a device model [Devb]
4. Install Assistant SDK and samples [Sdk]

1.3 Circuit requirements

The following parts are required to build the circuit:

- a small sized breadboard
- one LED
- one 330 Ohm resistor
- two jumper wires

The LED can be wired as shown by Figure 1.1.

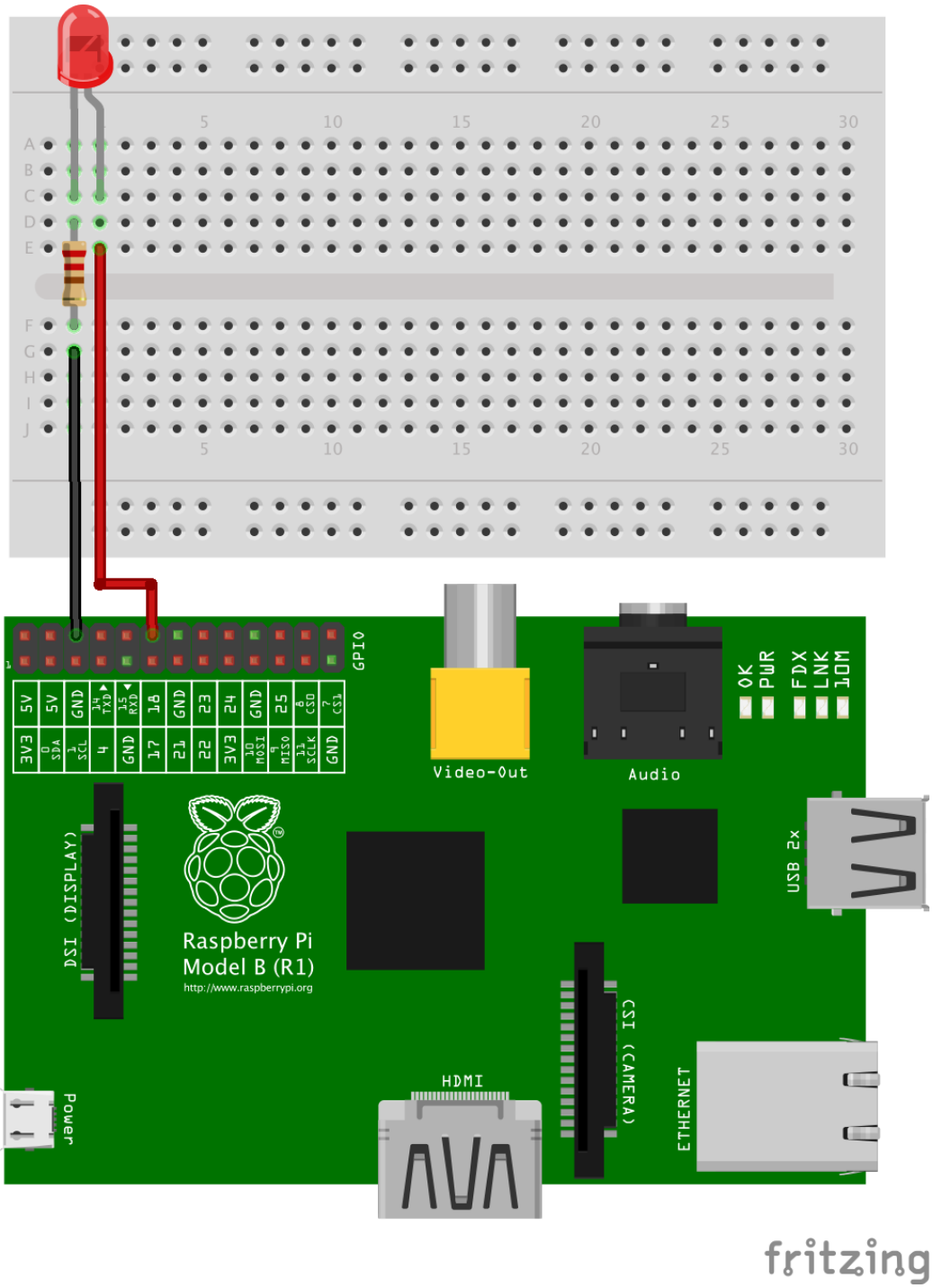

Figure 1.1: Wiring an LED to Raspberry Pi

1.4 Registering device custom actions

This section describes how to define a custom set of traits. The special trait we propose to support will sound like "increase brightness by 30 percent".

Google Assistant SDK docs [Cus] describes the following key items to be specified:

1. The custom pattern to be matched against the user query
2. The action associated with that query pattern
3. The response spoken back to user
4. The custom command sent to the device, including the extracted parameters from the query

The custom action is defined using the so called Google Action Package.

1.4.1 Action Package Definition

The action package you will use is already defined in *action.json* file. This file contains the specification of brightness increase action:

```
{
    "manifest": {
        "displayName": "LED brightness",
        "invocationName": "LED brightness",
        "category": "PRODUCTIVITY"
    },
    "actions": [
        {
            "name": "ro.upt.ms.actions.increase_brightness",
            "availability": {
                "deviceClasses": [
                    {
                        "assistantSdkDevice": {}
                    }
                ]
            },
            "intent": {
                "name": "ro.upt.ms.intents.increase_brightness",
                "parameters": [
                    {
                        "name": "number",
                        "type": "SchemaOrg_Number"
                    }
                ],
                "trigger": {
                    "queryPatterns": [
                        "increase brightness by $SchemaOrg_Number:number percent"
                    ]
                }
            },
            "fulfillment": {
                "staticFulfillment": {
                    "templatedResponse": {
```

```
                    "items": [
                        {
                            "simpleResponse": {
                                "textToSpeech": "Increasing the brightness by $number percent"
                            }
                        },
                        {
                            "deviceExecution": {
                                "command": "ro.upt.ms.commands.increase_brightness",
                                "params": {
                                    "number": "$number"
                                }
                            }
                        }
                    ]
                }
            }
        }
    ]
}
```

The *intent* field defines the voice query format and the parameters to be extracted. The *fulfillment* field describes the response spoken back to the consumer and which information to be sent to the target device.

1.4.2 Action Package Deployment

You've seen how to define custom actions using the Action Package format, but we still have not registered the action with the Google Assistant API. For this particular step we use the *gactions* tool, so let's download it alongside the *actions.json* file:

```
$ cd assistant-sdk-python/google-assistant-sdk
$ wget https://dl.google.com/gactions/updates/bin/linux/arm/gactions
```

Assure that *gactions* is executable:

```
$ chmod +x gactions
```

Execute the following command which registers the custom action in the context of your development project:

```
$ ./gactions update --action_package actions.json --project project-id
```

where *project-id* is the identifier which was previously generated in project setup step

Finally, deploy your action package into test mode:

```
$ ./gactions test --action_package actions.json --project project-id
```

1.5 Extending the provided sample

The script we'll modify in this section is included within SDK sample code.

Our repository already contains the sample code, therefore we just need to navigate there:

```
$ cd assistant-sdk-python/google-assistant-sdk/googlesamples/assistant/library/
```

Now, open the *hotword.py* file with an editor of your choice:

```
$ nano hotword.py
```

The first code section we'll discuss can be found under *process_event()* function:

```
if event.type == EventType.ON_DEVICE_ACTION:
        for command, params in process_device_actions(event, device_id):
            print('Do command', command, 'with params', str(params))

            if command == "ro.upt.ms.commands.increase_brightness":

                percentage = int( params['number'] )

                global duty_cycle, pwm

                # TODO: compute new percentage and then pwm's change duty cycle
```

Firstly, we react only when the *increase_brightness* action was sent to our device. Only then we extract the percentage value and set the new value to the PWM object.

The second code section shows how the initialization of the PWM is executed. You can find it under the *main()* funtion:

```
with Assistant(credentials, args.device_model_id) as assistant:
        events = assistant.start()

        print('device_model_id:', args.device_model_id + '\n' +
              'device_id:', assistant.device_id + '\n')

        # TODO: set up the PWM object which controls the LED
```

Finally, it's time to execute the script:

```
$ sudo python hotword.py --device_model_id my-model
```

where *my-model* is the identifier which was previously generated in model registration step

Let's try our query:

"Ok Google, increase brightness by 10 percent"

2. Assignments

1. Complete TODO items contained by *main()* and *process_event()* functions.

2. Extend the provided sample to support the following user query:
   "Ok Google, decrease brightness by x percent", where x is variable.

3. Bibliography

3.1 References

[Aud] *Audio configuration*. https://developers.google.com/assistant/sdk/guides/library/python/embed/audio?hardware=rpi. [Online; accessed April-2018]. 2018 (cited on page 5).

[Deva] *Configure the developer project*. https://developers.google.com/assistant/sdk/guides/library/python/embed/config-dev-project-and-account. [Online; accessed April-2018]. 2018 (cited on page 5).

[Cus] *Custom trait definition docs*. https://developers.google.com/assistant/sdk/guides/library/python/extend/custom-actions. [Online; accessed April-2018]. 2018 (cited on page 7).

[Devb] *Register the device model*. https://developers.google.com/assistant/sdk/guides/library/python/embed/register-device. [Online; accessed April-2018]. 2018 (cited on page 5).

[Sdk] *SDK instalation*. https://developers.google.com/assistant/sdk/guides/library/python/embed/install-sample. [Online; accessed April-2018]. 2018 (cited on page 5).

3.2 Image credits

- First page illustration.
  http://www.northeastern.edu/levelblog/2018/01/25/guide-iot-careers
- Chapter header background.
  https://blogs.microsoft.com/iot/2015/03/17/simplifying-iot/