Control LEDs with your voice

Controlling the world with Google AIY

The Google AIY Voice Kit came free with the May 2017 print issue of the MagPi, so well done if you have gotten your hands on one. If you haven't yet managed to acquire a Voice Kit, you can join the mailing list to receive a notification when they become commercially available by signing up here (https://docs.google.com/forms/d/e/1FAIpQLSev7IQBFUaDIv5tx1Decxd5Ya5AqYSEvD72hJySeaRDogaqAw/viewform?c=0&w=1).

What you will learn

- How to connect an LED to the AIY Voice Kit
- How to extract information from voice commands
- How to trigger the GPIO pins using voice commands

This resource covers elements from the following strands of the Raspberry Pi Digital Making Curriculum (https://www.raspberrypi.org/curriculum/):

- Combine programming constructs to solve a problem (https://www.raspberrypi.org/curriculum/programming/builder)
- Combine inputs and/or outputs to create projects or solve a problem (https://www.raspberrypi.org/curriculum/physical-computing/builder)
- Use basic materials and tools to create project prototypes (https://www.raspberrypi.org/curriculum/manufacture/creator)

What you will need?

Hardware

- A Raspberry Pi computer
- A Google AIY Voice Kit
- An LED
- 2 x male-female jumper leads
- A 50-100Ω resistor

Software

- aiyprojects image (https://dl.google.com/dl/aiyprojects/voice/aiyprojects-latest.img.xz)

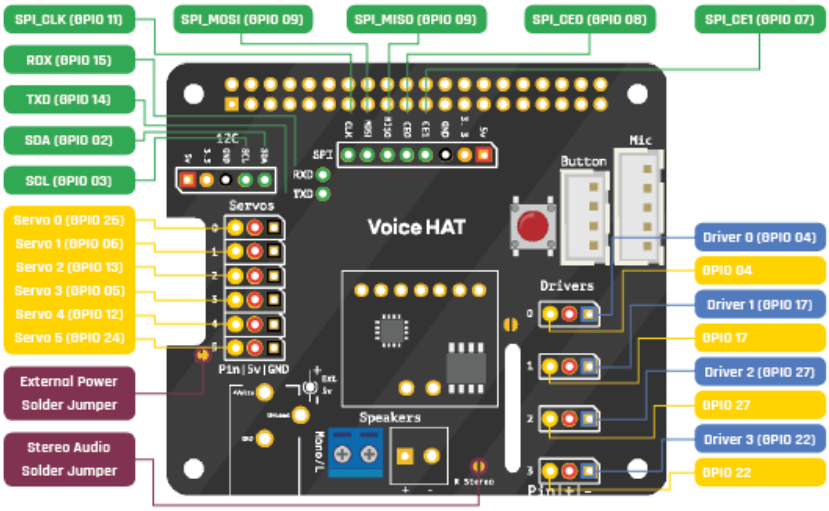
Solder on header pins

In this project, you're going to use the Voice Kit to make an LED blink in response to a voice command. If you can make an LED, then there really are very few limits to what you can control.

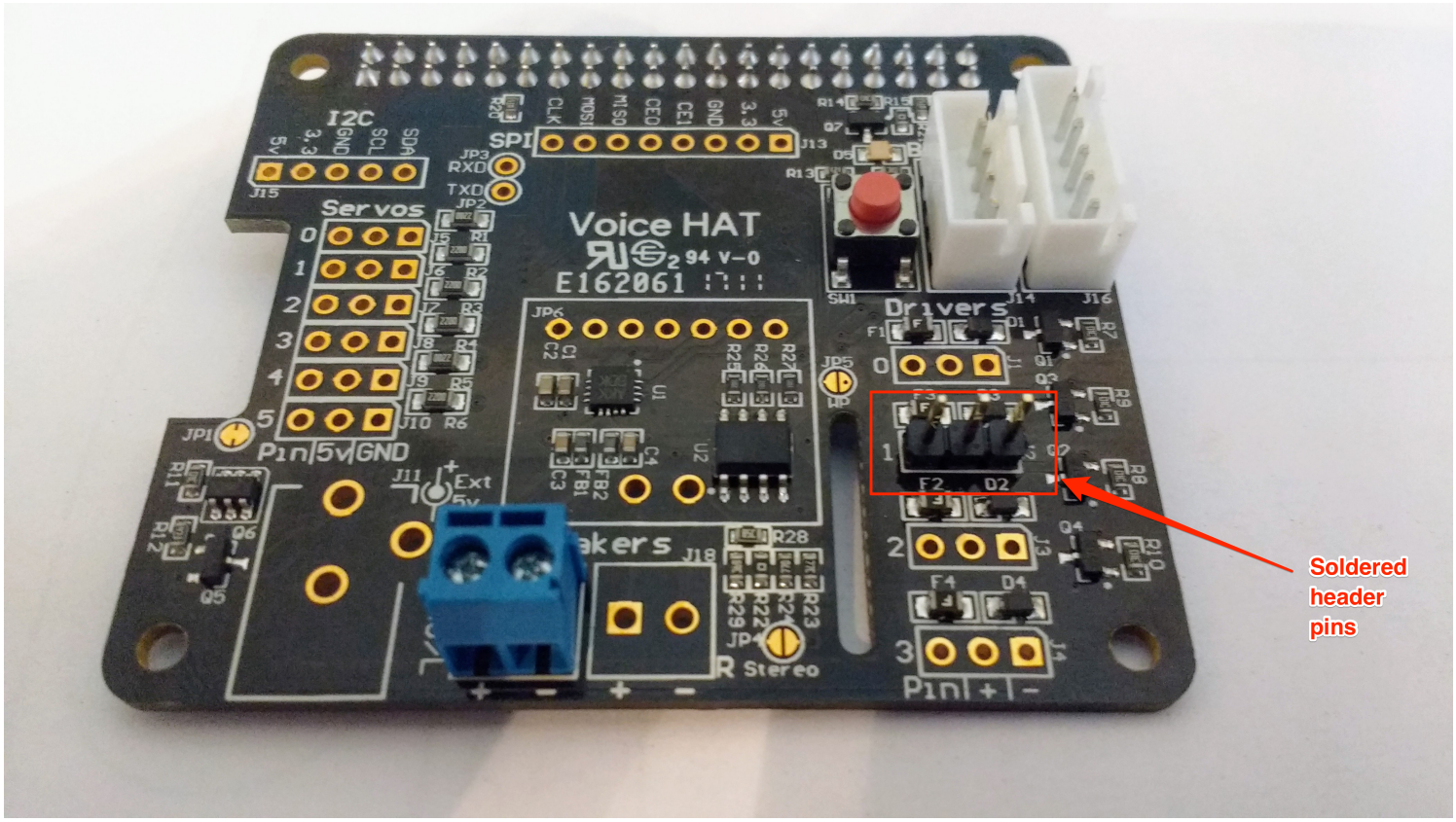
The first thing to do is to set up the Voice HAT. As you will be controlling an LED, you will need to use some soldered header pins to allow you to access the GPIO pins of the Raspberry Pi.

You can solder a set of three header pins to the holes on the board that are in the column of **Drivers**. In particular, you want row **1**.

You can see the mapping of all the GPIO pins on the following schematic, in case you want to use a different GPIO pin.



And here is a photograph showing the three soldered header pins:



If you have never soldered before, and need some help, then have a look at our Getting started with soldering (https://www.raspberrypi.org/learning/getting-started-with-soldering/) guide, or watch the video below.

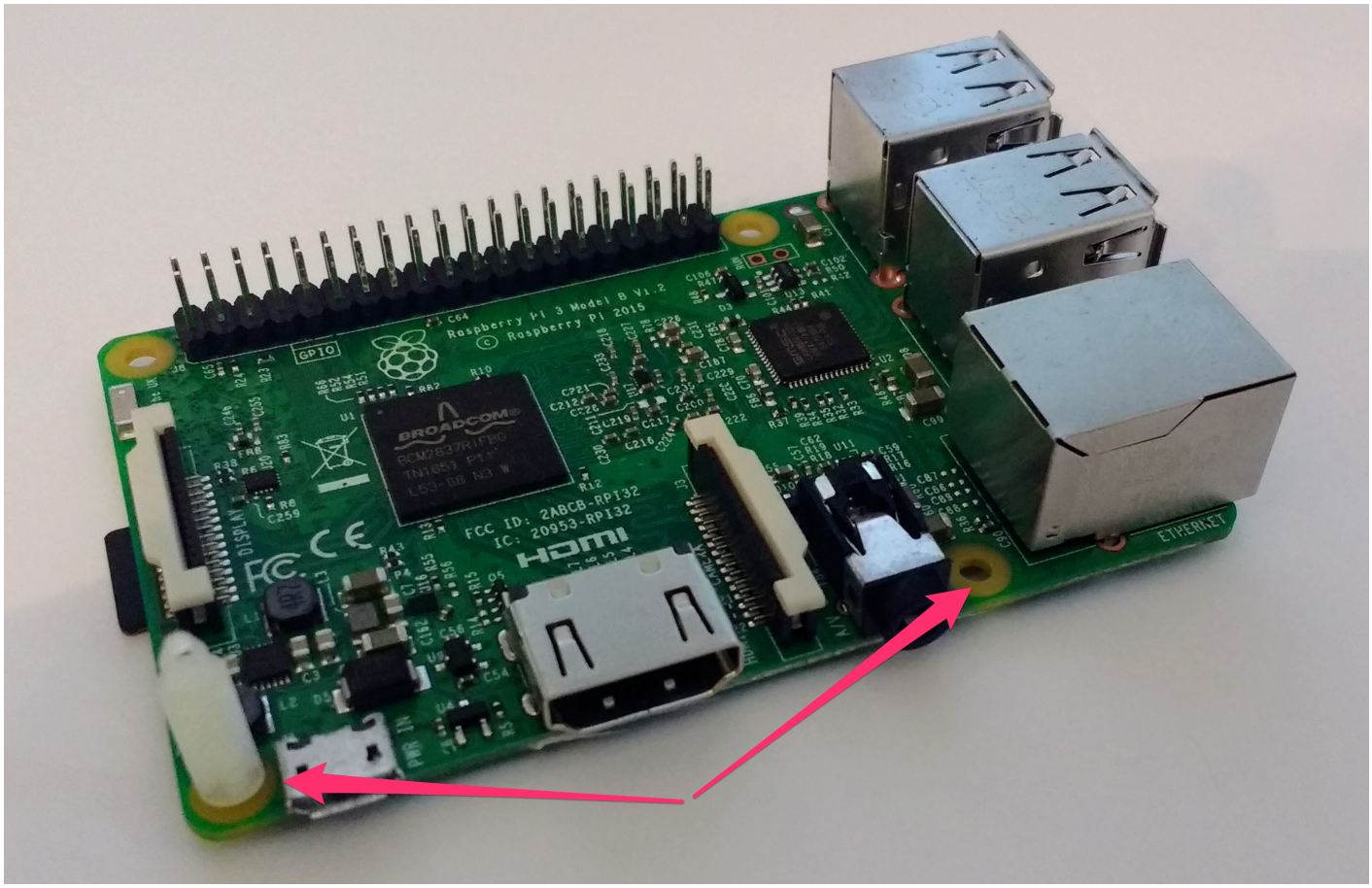
Getting started with soldering

Setting up the hardware

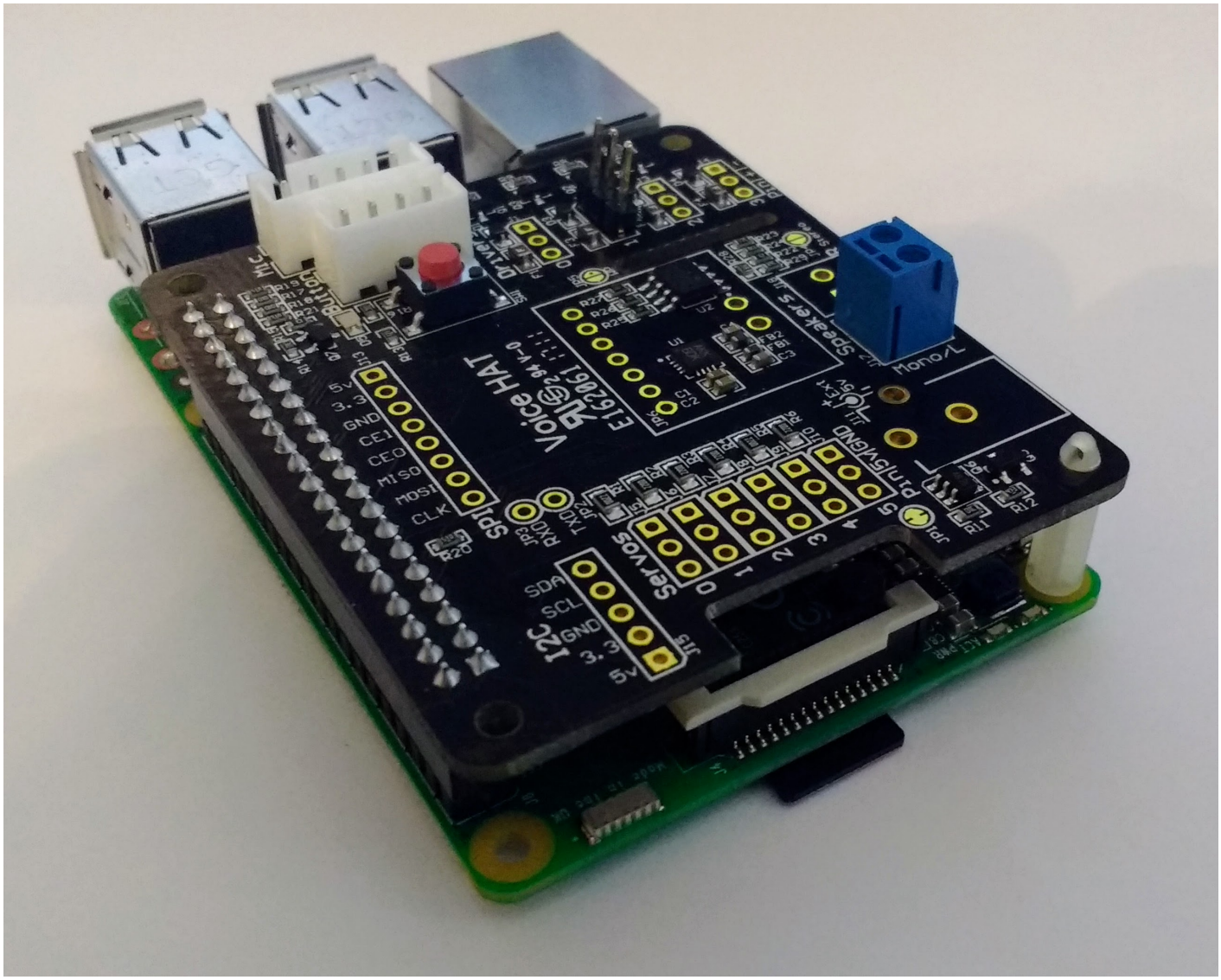
You can follow the build guide on the Google AIY website (https://aiyprojects.withgoogle.com/voice/#assembly-guide-4-put-it-all-together) if you want. However, it uses the cardboard box to house the kit, and this will restrict access to the GPIO pins. If you want to follow a simpler guide, then use the instructions below.

Assemble the AIY Voice Kit

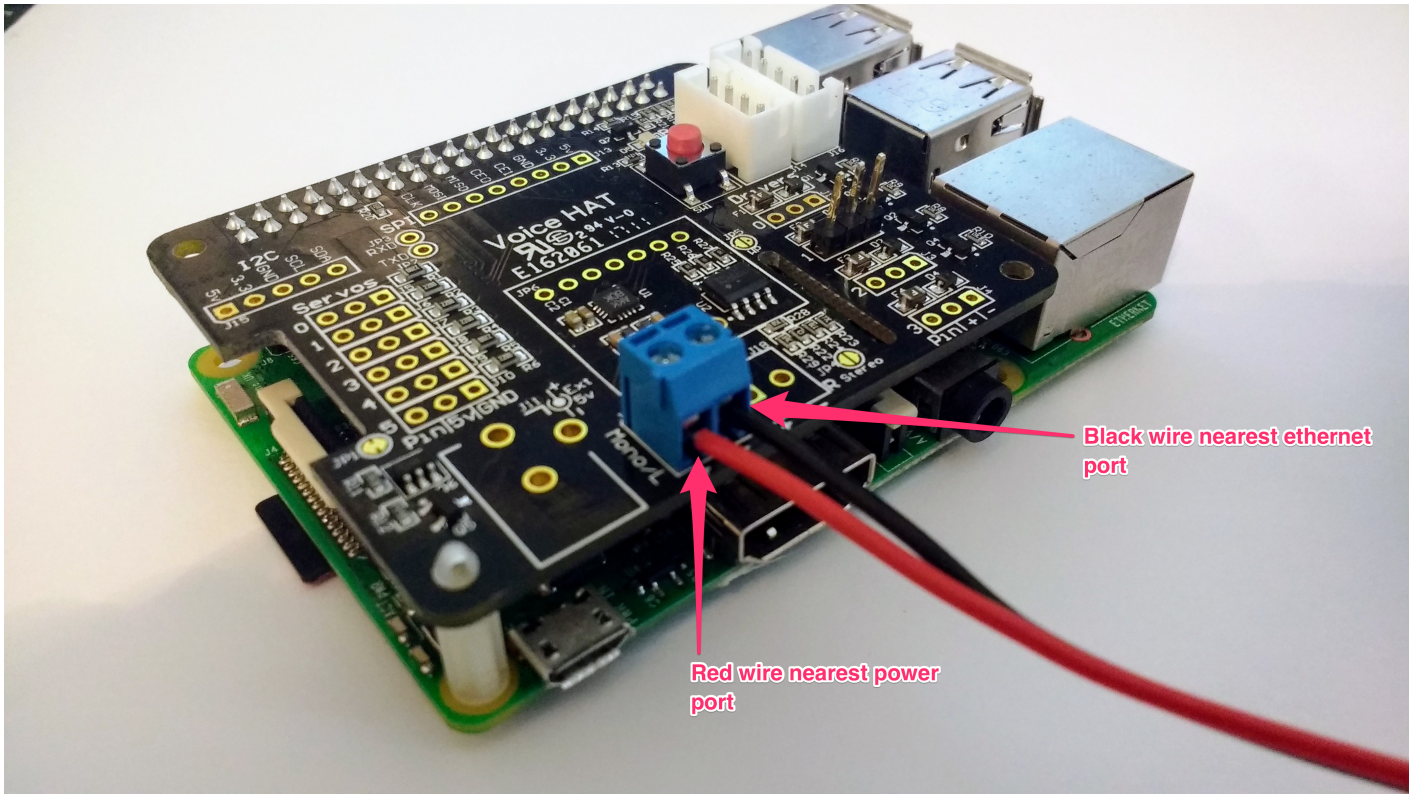
- First, you need to use the plastic **standoffs** to help support the Voice Kit HAT when it is attached to the Raspberry Pi. Insert the standoffs into the mounting holes opposite the GPIO pins.



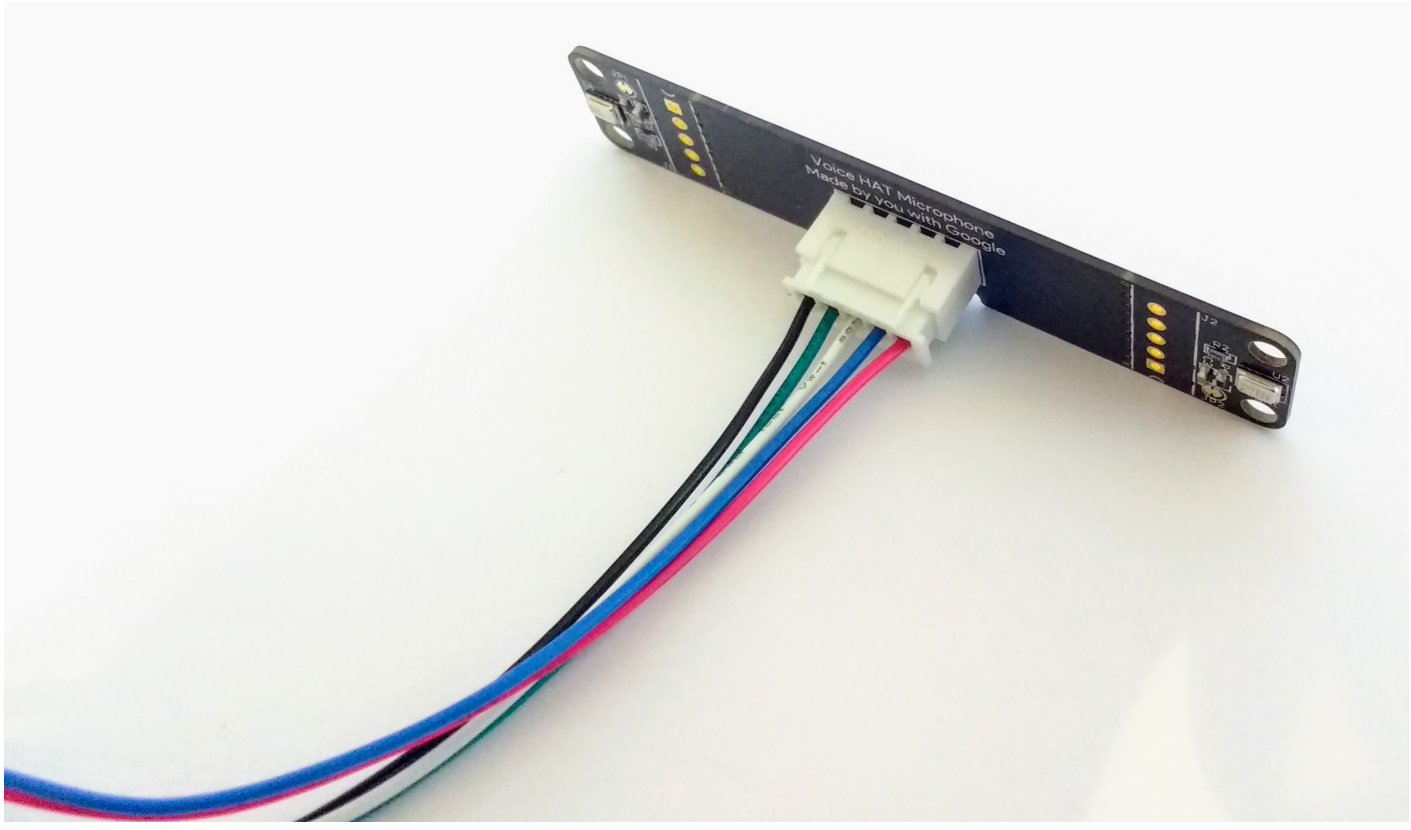
- You can now place the HAT onto the Raspberry Pi - make sure that the pins are all aligned.



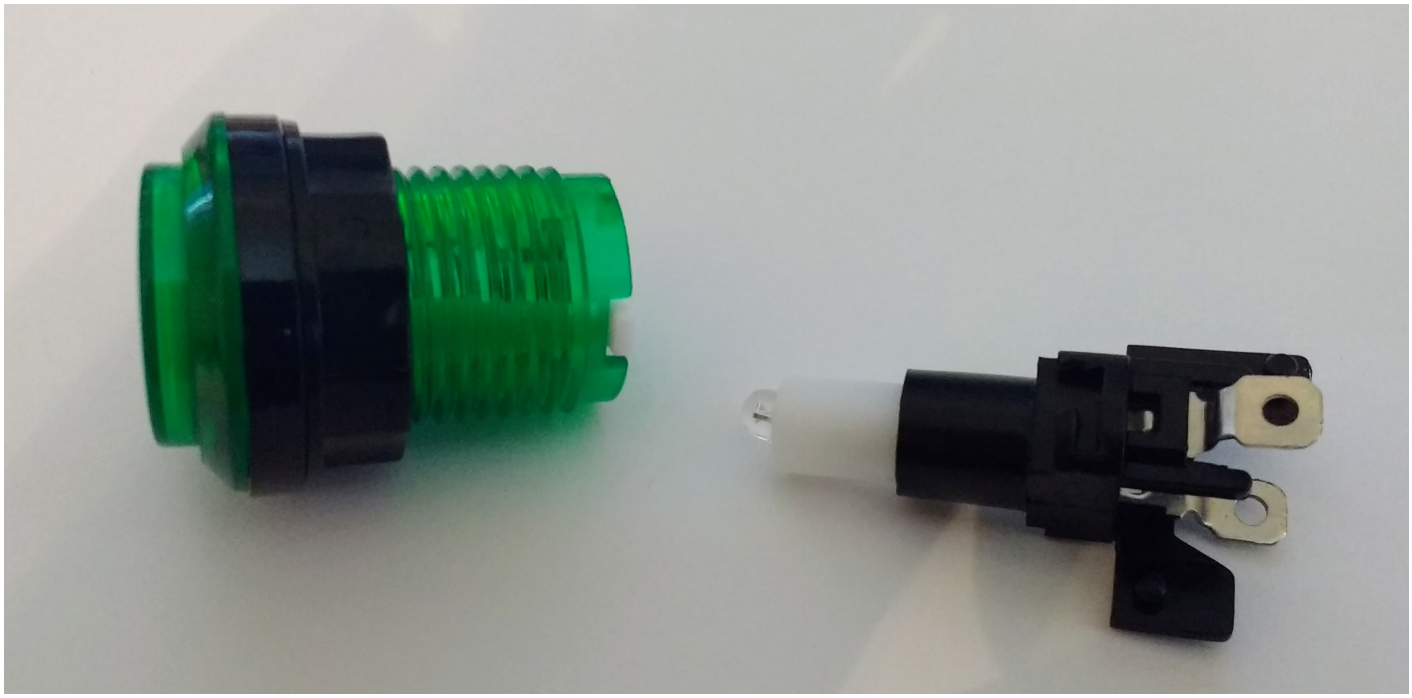
- Next, attach the speaker to the kit. It has to be wired in a particular way: the red wire needs to be inserted into the hole closest to the Raspberry Pi's Ethernet port. The black wire goes into the other hole. Use a Phillips-head screwdriver to secure the wires in place.



- Now it's time to connect the microphone to its leads. The connectors only fit one way, so this shouldn't be too difficult.

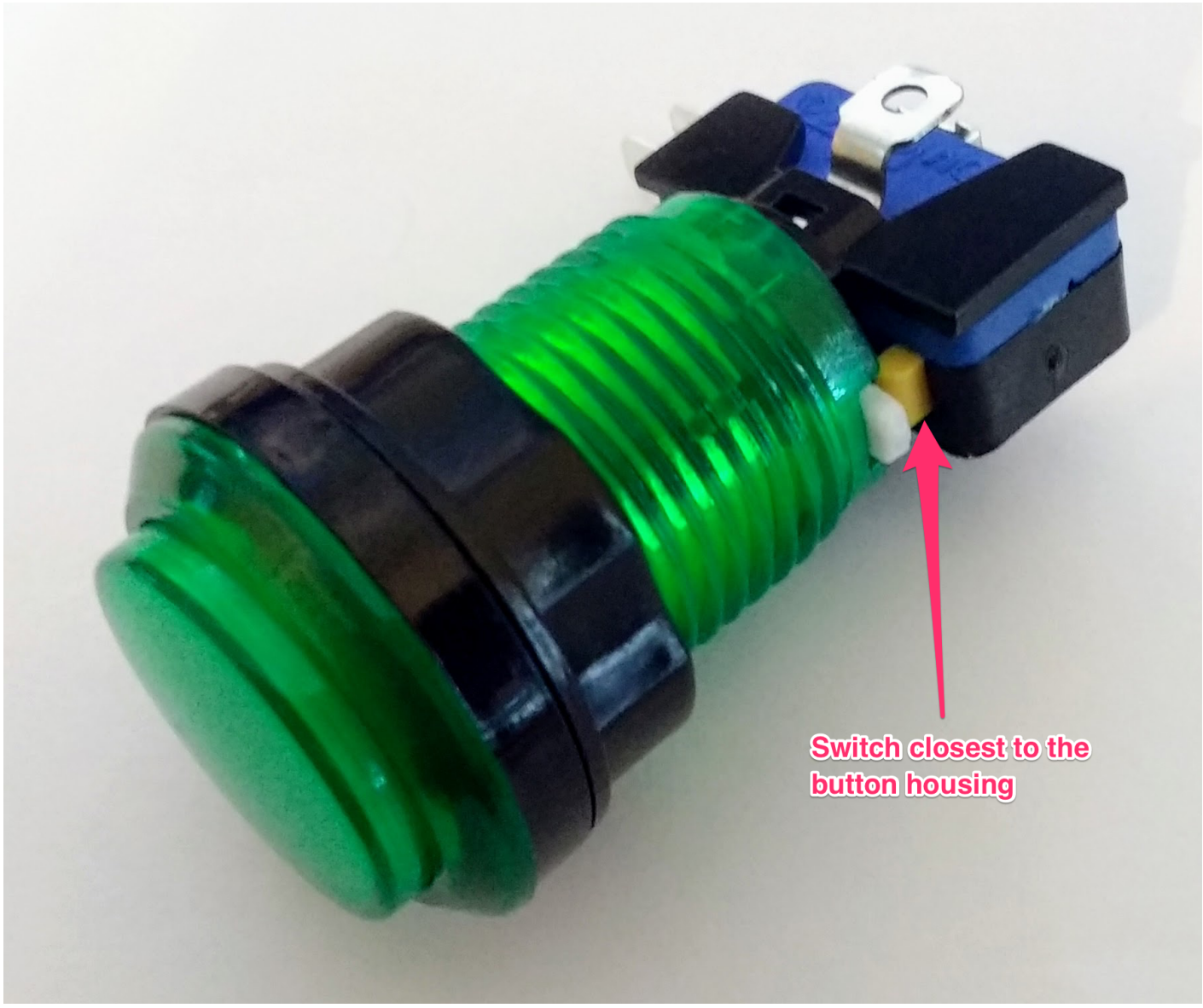


- The trickiest part is assembling the button. You'll need the button and the LED housing to begin with.

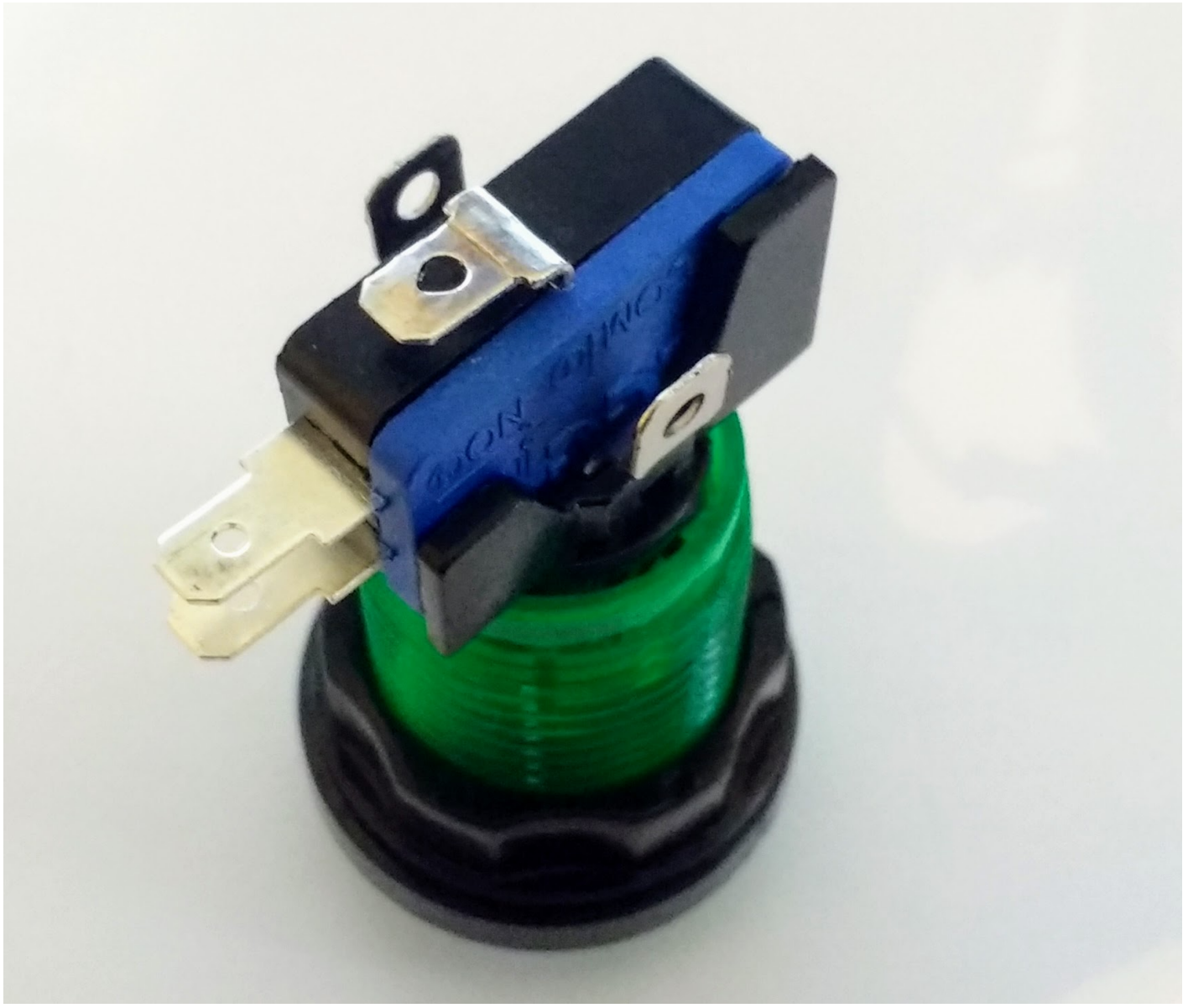


- Insert the LED housing into the button, and then twist it to secure it in place.

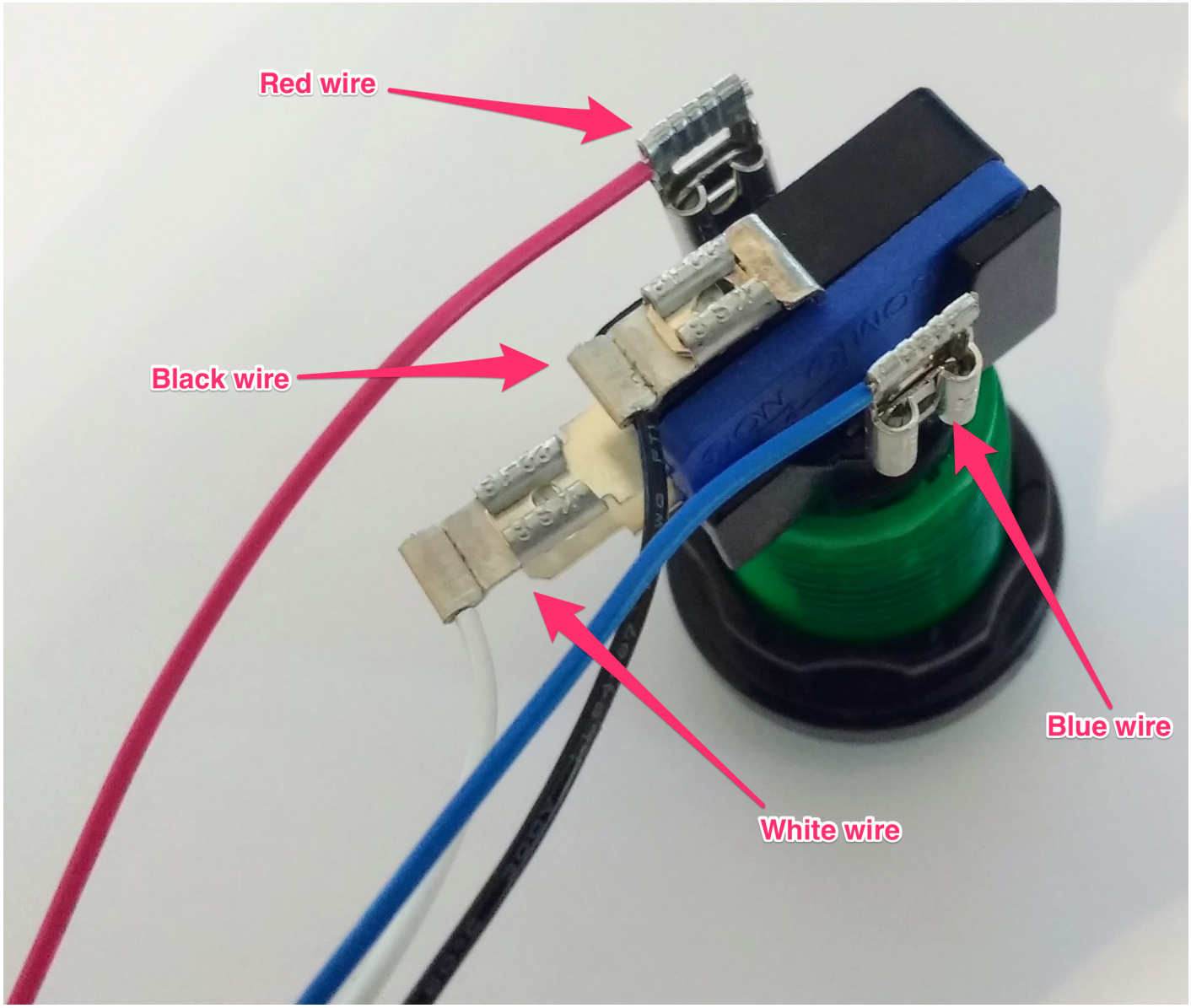
- Then the switch needs attaching. This can be awkward. The holes of the switch need to align with the pegs on the LED housing. Just make sure that the small switch (here in yellow) is positioned closest to the button.



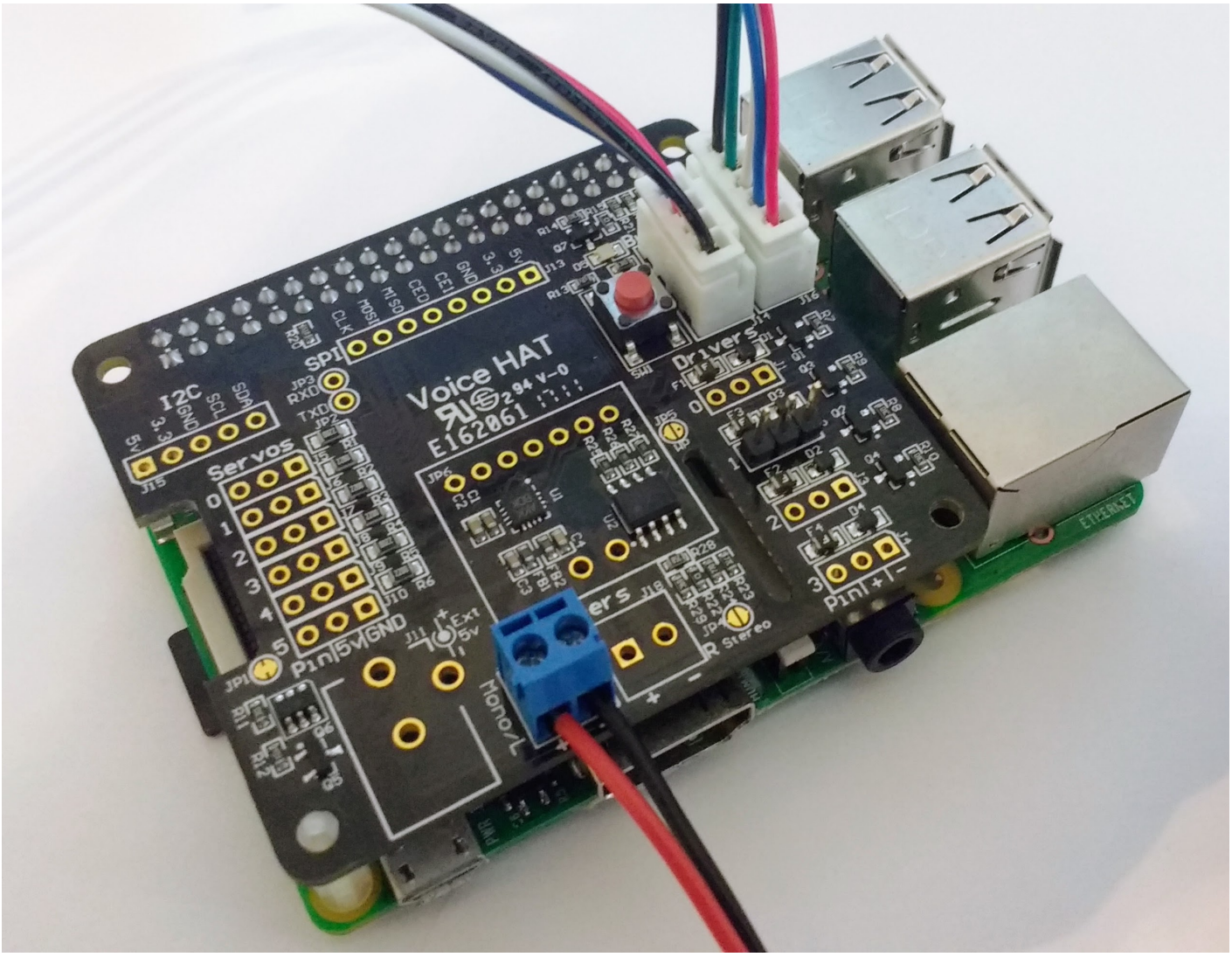
- Now you can attach the leads to the button.



- Attach the leads as shown in the image below.



- To finish, attach the microphone and the button to the HAT as shown.



Install the Software

If you like, you can install the software for the Voice Kit manually. Google provides this guide (https://aiyprojects.withgoogle.com/voice/#makers-guide) to take you through the process. It is far easier, however, to use their image on an SD card.

You can download their image here (https://dl.google.com/dl/aiyprojects/voice/aiyprojects-latest.img.xz), and if you want guidance on how to burn an image to an SD card, have a look at our guide here (https://www.raspberrypi.org/learning/software-guide/quickstart/).

The Google image comes as an `.xz` file. To extract this on Linux, you can install `unxz`.

```
sudo apt update && sudo apt install zx-utils -y
unxz aiyprojects-2017-05-03.img.xz
```

On Windows or MacOS, **Etcher** (https://etcher.io/) should handle this for you.

Then just insert your SD card and boot your Raspberry Pi. Your button should be slowly pulsing and your desktop should look like this:

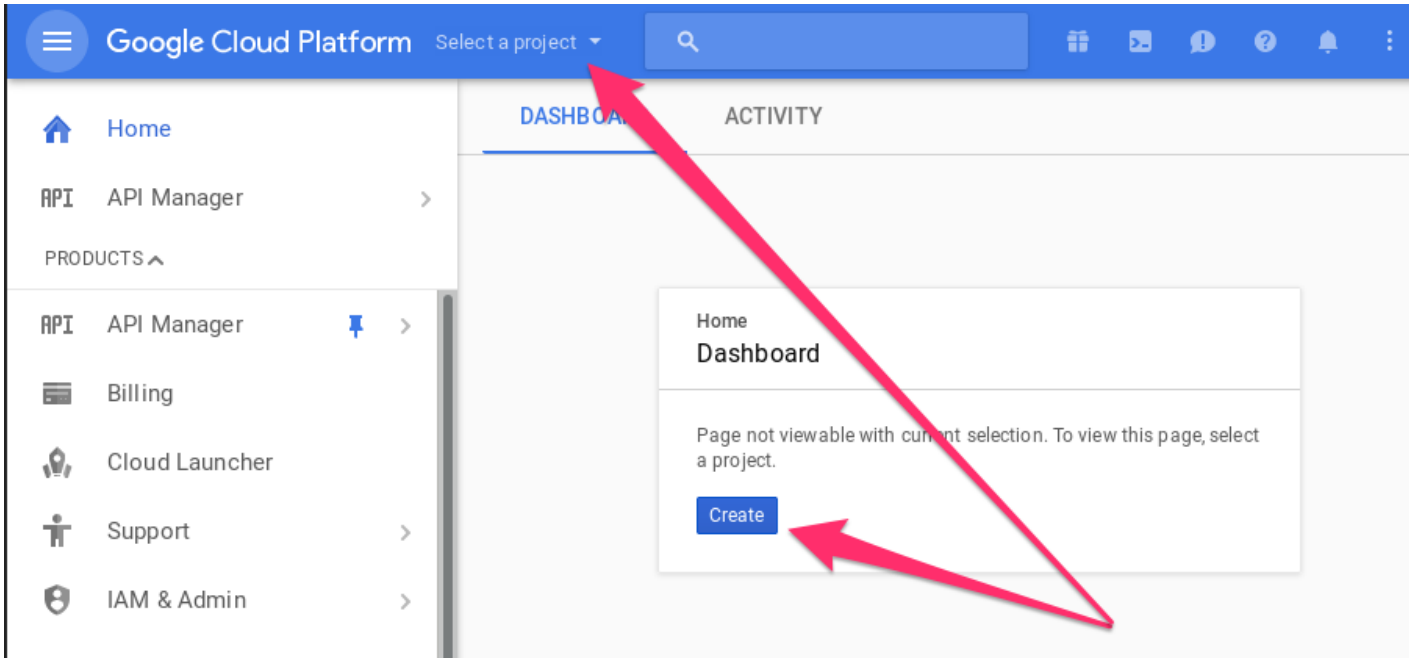
Setting up the Assistant API

Once your Raspberry Pi has booted, you're going to need some credentials from Google for the kit to work. Follow the steps below to enable the Google Assistant API.

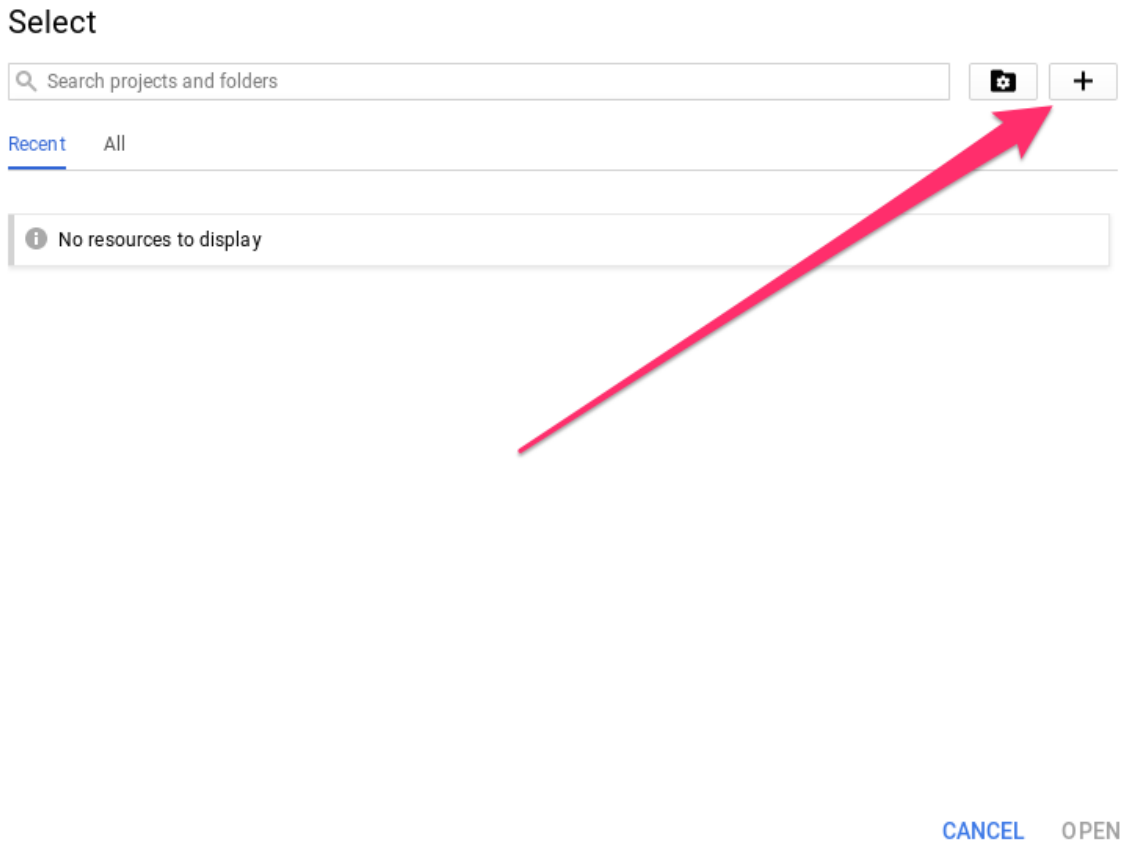
Register the Google Assistant API

- Navigate to Google's Cloud Platform (https://console.cloud.google.com/home/dashboard?project=ayi-led) and ensure you are logged into your Google account.

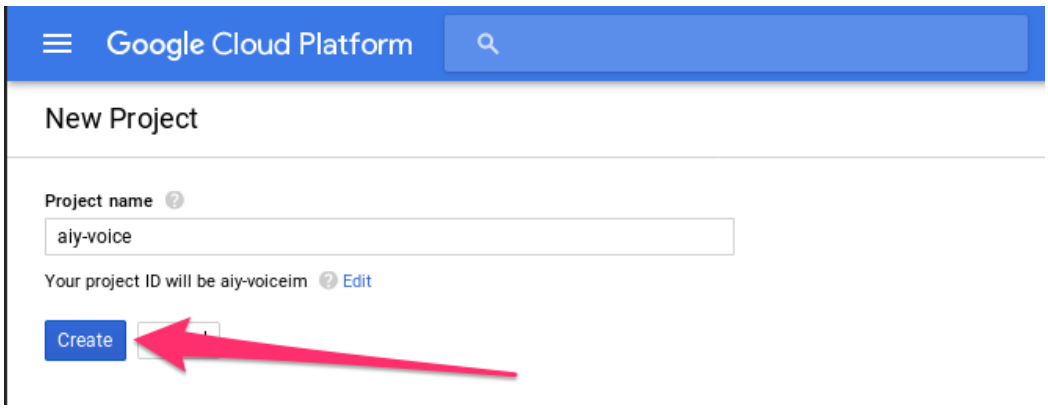
- You need to create a new project to get started. If you have no existing projects, click on the **Create** button. If you do have existing projects, you can click on the **Select a project** drop-down menu.



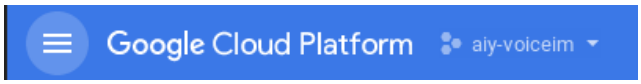
- If you clicked the drop-down menu, you can click on the **+** symbol to create a new project.



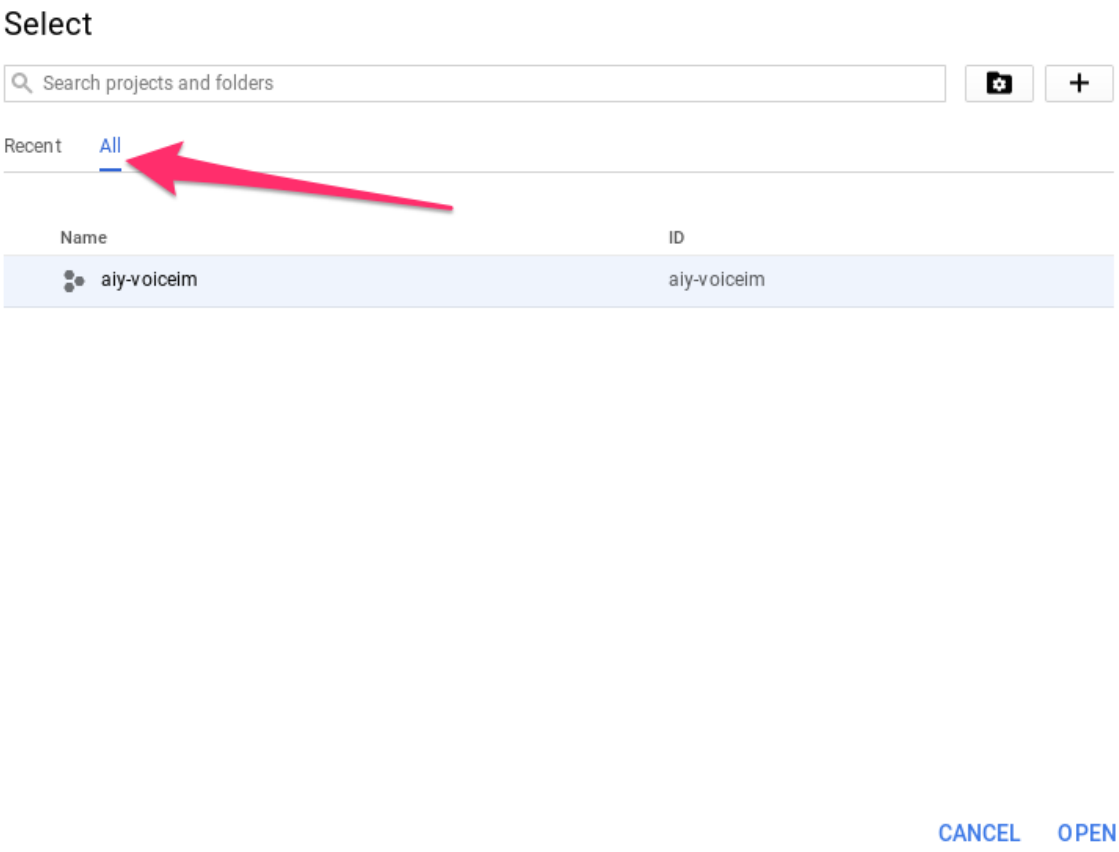
- Give your project a name (it doesn't matter what you call it), and then click on **Create**.



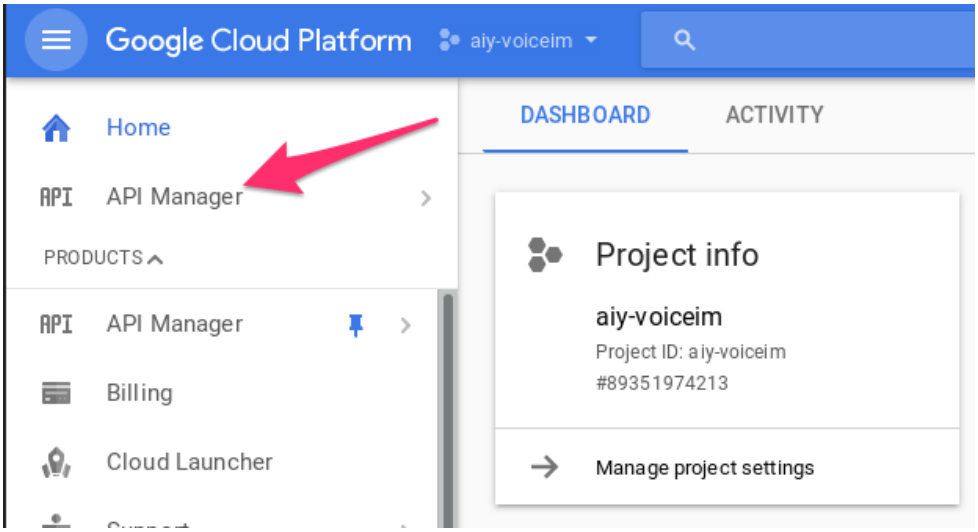
- Check that your project appears in the drop-down menu, as shown below.



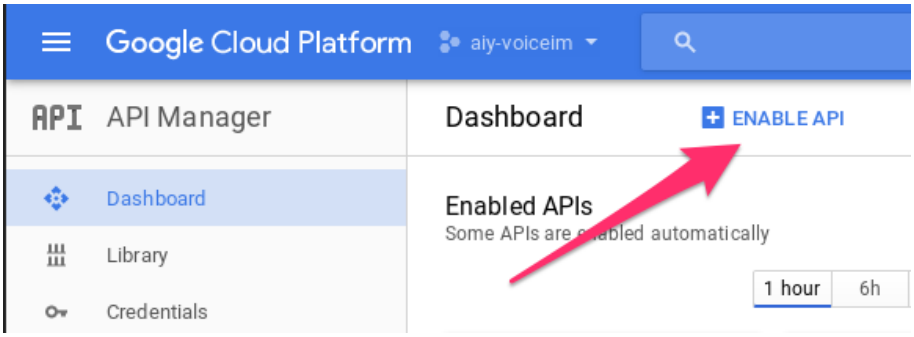
- If it doesn't, click on the drop down and then select **All** from the popup to find and select your project.



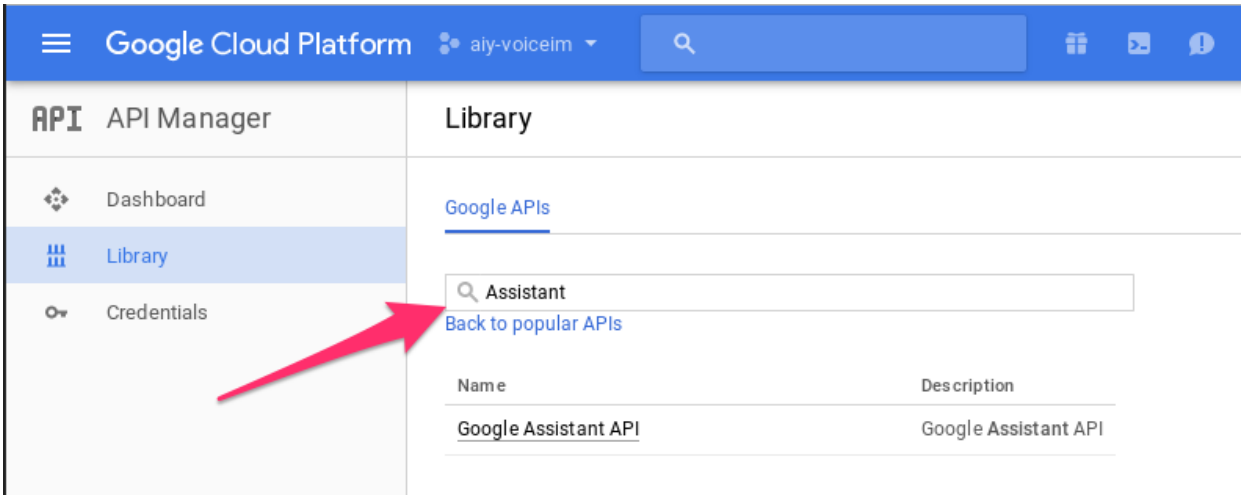
- Now you need to click on the **API Manager**.



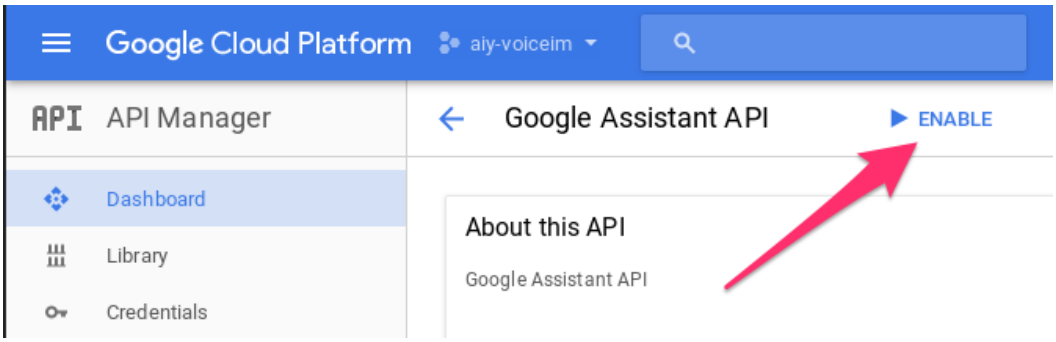
- Then click on the **ENABLE API** link.



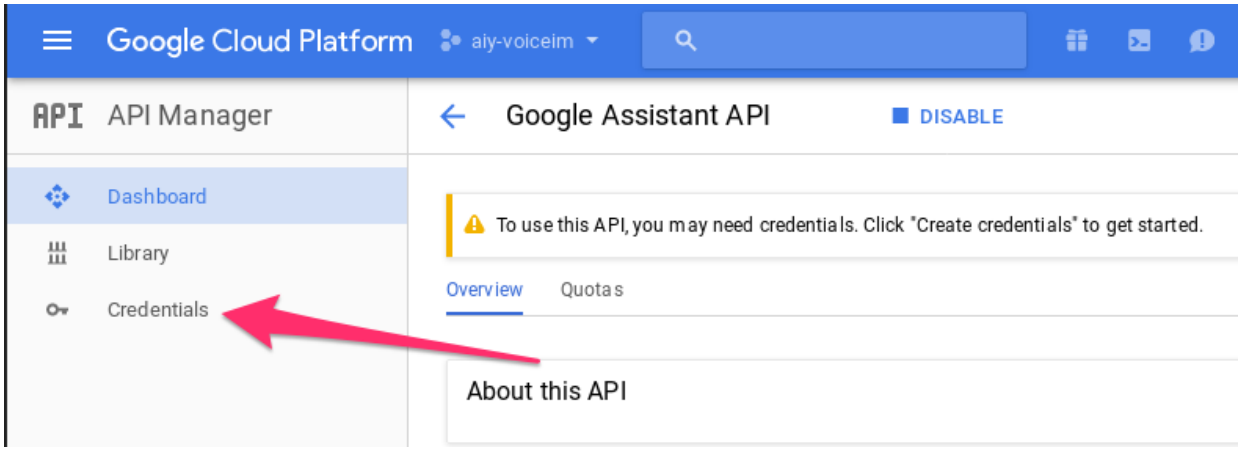
- Use the search box to find the **Google Assistant API**.



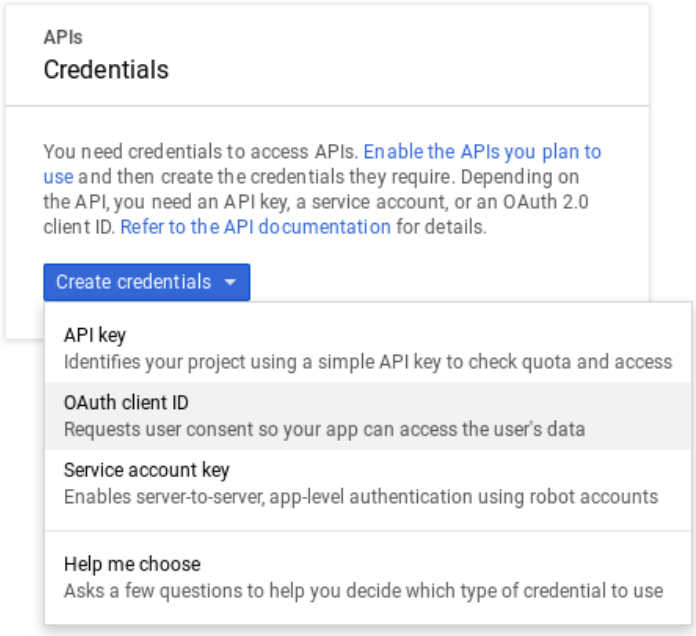
- Once you have selected the API, click on the **ENABLE** link.



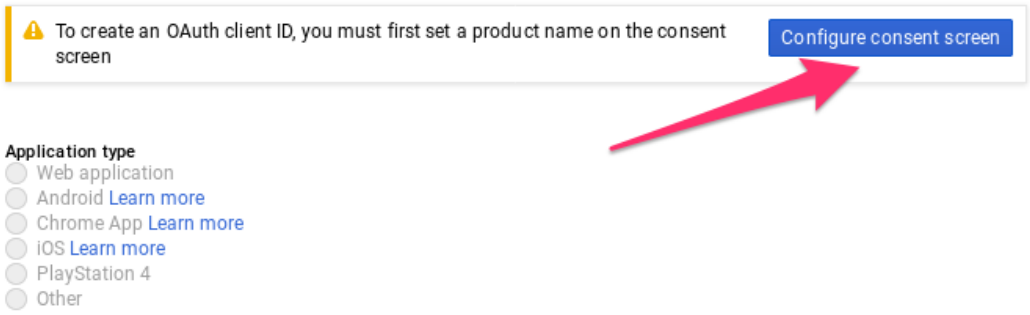
- Now, click on the **Credentials** link in the side bar to create some new credentials.



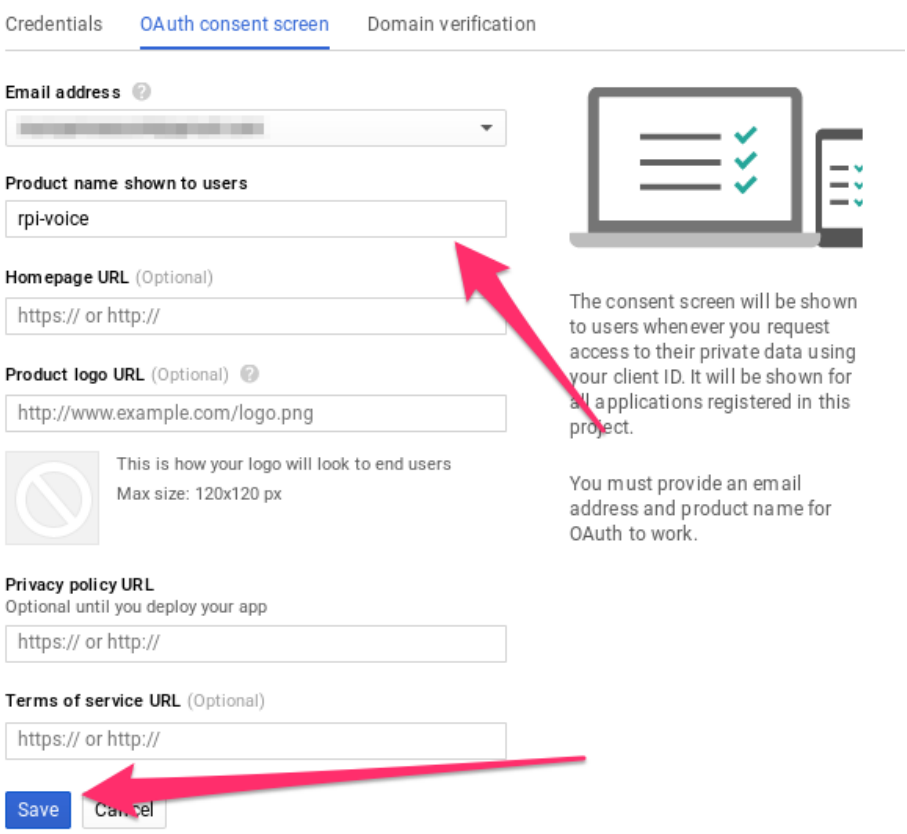
- You can create your credentials by selecting **OAuth client ID** from the drop-down menu.



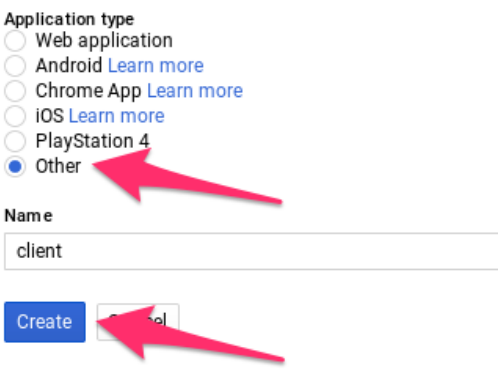
- You're now going to need to configure the consent screen.



- You can type your email address and project name into the boxes and leave the other fields blank if you like. Then click **Save**.



- On the main screen, you can now choose **Other** as your application type and give your application any name you choose. Then click **Create**.



- You should be presented with your **client ID** and **client secret**. You can just click on **OK** though, as you're going to download a **JSON** file containing these details.

- Click on the **Download** icon to download your secrets to your computer, and then you're finished.

The **secrets** file that you downloaded will be called something like `client_secret_89351974213-jsno1i2s7lu9mv4q9bjbf3pas6cpnbe5.apps.googleusercontent.com.json`. You need to rename it `assistant.json` and place it in your `/home/pi` directory.

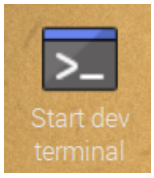
To do this, open a terminal and type:

```
cd ~/
mv Downloads/client_secret* assistant.json
```

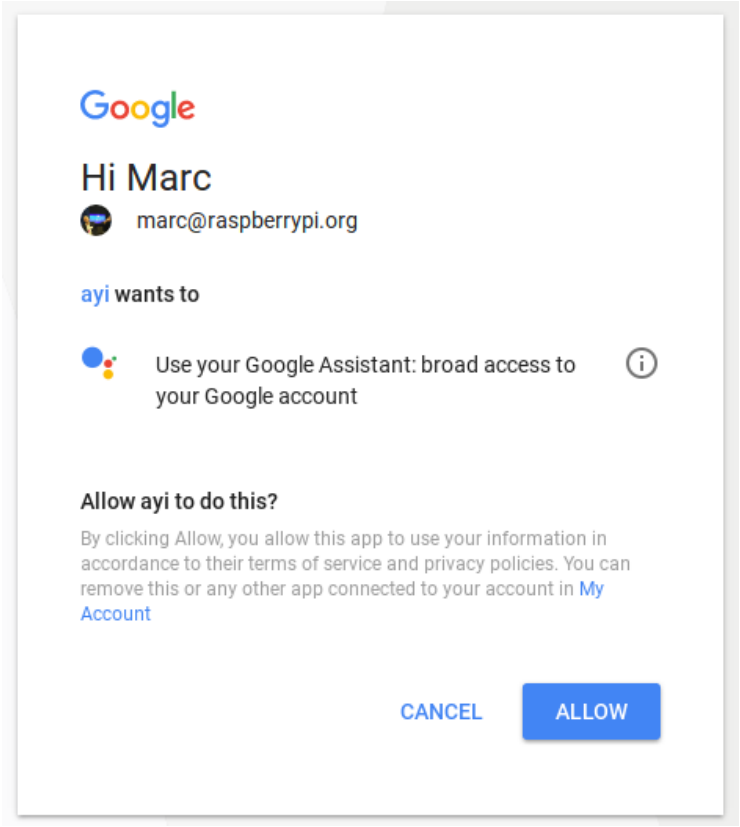
Test it's working

With the hardware and software all set up, you need to test that your Voice Kit is working.

- Click on the **Start dev terminal** icon on the desktop to open a terminal window.



- To start the **Voice Kit** program manually, you can simply type `src/main.py` into the terminal.
- If it is your first time running this program, **Chromium** will open and ask you to login and authorise the use of the Google API.



- Click **ALLOW** to enable access to the API. Now you should be able to use the button to begin capturing your voice commands. There are several built-in instructions you can use. Try pushing the button and then saying any of the following phrases:
    - "What are the three laws of robotics?"
    - "What is the time?"
    - "IP Address"
- You can also ask it questions that will result in a simple Google search, for example:
    - "Who is the Prime Minister?"
    - "What is the air-speed velocity of an unladen swallow?"
    - "What is the air-speed velocity of an unladen African swallow?"
- Have a good play with the device before learning how to hack it to create your own voice commands.

Simple voice responses

The AIY Voice Kit software allows you to add your own simple voice commands that will result in simple responses.

- Using a text editor or IDLE (**Menu** –> **Programming** –> **Python 3 (IDLE)**, open the file called `action.py`. You can find it in `/home/pi/voice-recognizer-raspi/src/action.py`.

- Most of this file consists of instructions on how to use the kit, but if you scroll down, you will eventually come to the following comments:

```
# ========================================
# Makers! Add your own voice commands here.
# ========================================
```

- Here's where you can add some simple voice commands and the response you would like to receive back. Below the comment, you can now add your own actions. Try adding the following lines - make sure that you keep the indentation.

```
# ========================================
# Makers! Add your own voice commands here.
# ========================================

actor.add_keyword("what's up", SpeakAction(say, "I'm fine, thank you"))
```

- What does this line do? `actor.add_keyword("what's up"` instructs the code to listen for the keywords **"what's up"** spoken by the user. `SpeakAction(say, "I'm fine, thank you")`, instructs the program to respond with the words `"I'm fine, thank you"`.

- Have a go at running this code, and test that it's working. You'll need to go back to the terminal window, press `Ctrl + C` if the progam is currently running, and then type `src/main.py` to restart the Voice Kit software.

- Push the button and then ask the Voice Kit "What's up?"

- Now try adding your own set of keywords and responses below the one you have just written.

Making your own actions

Now it's time to make your own actions. As well as responding to particular commands, you can make the Raspberry Pi perform any action you like when it receives a specific command.

- Scroll up to the section with the following comment:

```
# ========================================
# Makers! Implement your own actions here.
# ========================================
```

- Below it you can add your own **actions**. An action is the thing that you want your Voice Kit to do. Below is about the most basic action you can come up with. Don't worry if you've never written a **class** before, as it can be kept fairly simple.

```
# ========================================
# Makers! Implement your own actions here.
# ========================================

class PrintHelloWorld():

    def run(self, voice_command):
        print("Hello World!")
```

- Within the class `PrintHelloWorld()` is a single function called `run()`. Functions inside classes are called **methods**. This method will be used automatically by the Voice Kit software.

- All this class will do is print `Hello World!` to the console when the `run` method is called. To make this happen, you need to add another voice command. Scroll back down to where you added the previous voice command and add in another `keyword`.

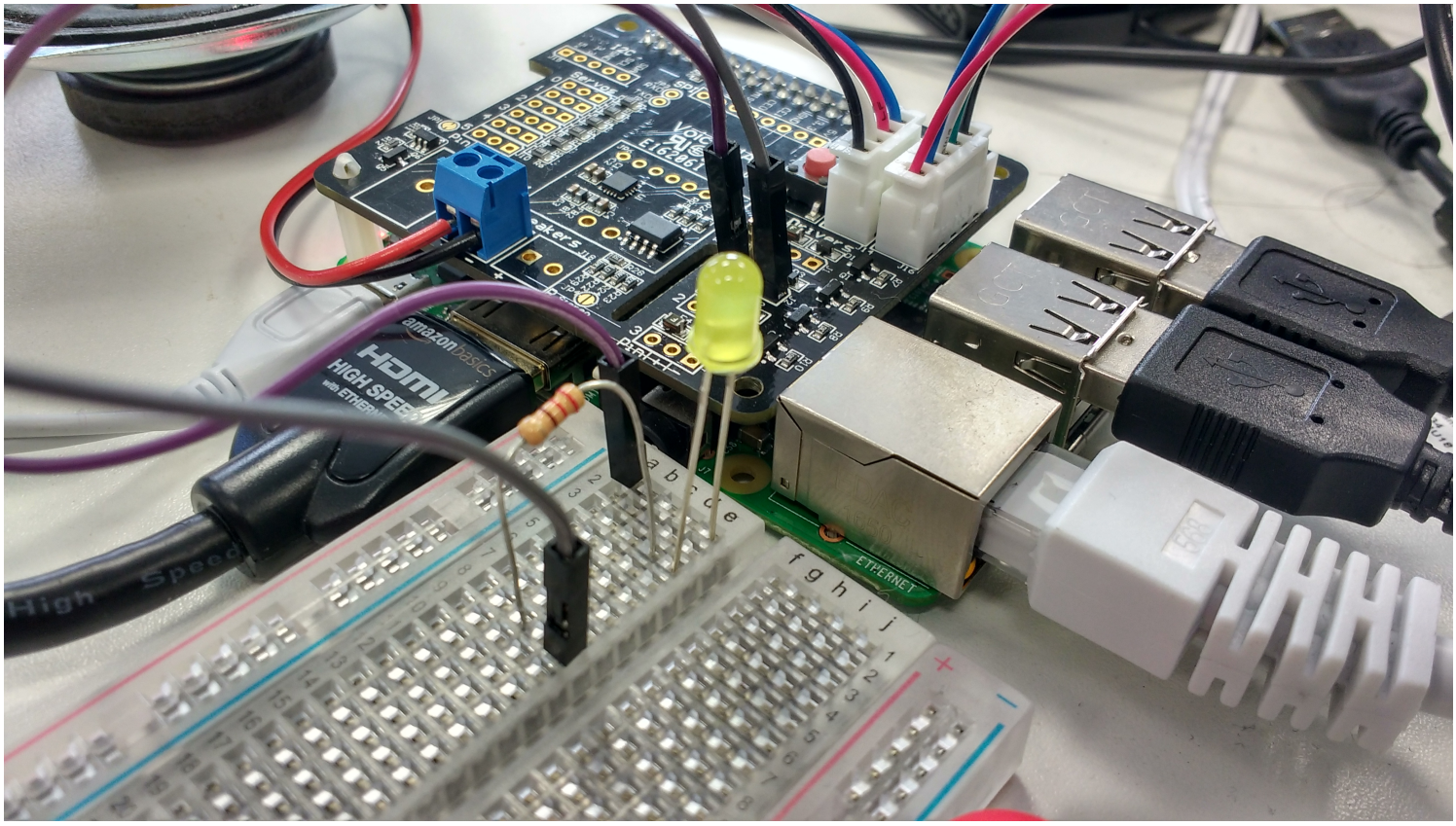
```
# =========================================
# Makers! Add your own voice commands here.
# =========================================

actor.add_keyword("what's up", SpeakAction(say, "I'm fine, thank you"))
actor.add_keyword("hello world", PrintHelloWorld())
```

- Now run the program in the teminal again (`src/main.py`) and push the button on the Voice Kit. If you now say “Hello world!”, you should see the output in the console.

Controlling an LED

Now is your chance to try and make an LED turn on and off again when a command is given.

- Firstly, connect an LED to the header pins you soldered on earlier.



- The positive (long) leg of the LED should be connected to the middle pin, and the negative leg (short leg) should be connected to the pin on the right of it.

- You’ll now need to do the following in the `action.py` file.
  - Near the top of the file, import the `LED` class from the `gpiozero` module
  - Create an `led` object on **GPIO 17**
  - Create a `ControlLED` class that turns the LED on, waits for 5 seconds, and turns the LED off again
  - Create a new voice command to trigger the class when the letters “LED” are spoken

- Have a look at the hints below if you're not sure how to proceed.

I need a hint

- Import the `LED` class and the `sleep` function near the top of the file, where the other imports are.
- Underneath this set up the `led` object:

```
led = LED(17)
```

- You can use the following code inside a `run` method to make the LED come on and off again.

```
led.on()
sleep(5)
led.off()
```

- The code for your voice command should look like this:

```
actor.add_keyword("LED", ControlLED())
```

- And here's what your class should look like:

```
class ControlLED():

    def run(self, voice_command):
        led.on()
        sleep(5)
        led.off()
```

Using the `say` function

It would be nice if you could get the class you have written to give some verbal feedback when the LED switches on and off again. To do this, you will need to use the special `say` function.

If you want to use the `say` function in your classes, it needs to be included in the `actor.keyword` call.

- First you can alter your voice command section of the script. The `ControlLED` class needs to use the `say` function.

```
actor.add_keyword('LED', ControlLED(say))
```

- Since you want to use some parameters with your `ControlLED` class, you need to make sure you're allowed to use them by amending the class's methods. You do this in the initialisation method.

```
class ControlLED():
    """Turns on an LED for 5 seconds"""

    def __init__(self, say):
        self.say = say
```

- Now within your `run` method, you can call `self.say` and pass it the string you want the Voice Kit to say.

```
class ControlLED():
    """Turns on an LED for 5 seconds"""

    def __init__(self, say):
        self.say = say

    def run(self, voice_command):
        self.say("Turning on LED")
        led.on()
        sleep(5)
        self.say("Turning off LED")
        led.off()
```

Using `voice_command`

Now that you can turn the LED on with a voice command, it would be nice if you could get it to stay on until another voice command turns it off again.

If you have a look at the `run` method you created, you should be able to see that it has a `voice_command` parameter.

```
def run(self, voice_command):
        self.say("Turning on LED")
        led.on()
        sleep(5)
        self.say("Turning off LED")
        led.off()
```

This `voice_command` that is automatically passed to your `run` method is a string translation of whatever the Google Assistant API thinks you said. (It makes mistakes sometimes.)

Here's what you're going to do:

- Within your run method, convert the `voice_command` string into all lower case
- Search through the string.
    - If it contains the word "on", turn the LED on
    - If it contains the word "off", turn the LED off
- You can have a look at the section below to learn how to find a specific sequence of characters within a string using Python.

Finding substrings with Python

Imagine you have a string (A sequence of characters) like "Programming with Python is awesome", and you want to find out whether it contains the word "with". How could you go about finding out?

- Python has a very handy operator called `in`. It can be used to find out whether a data structure such as a string (A sequence of characters) or a list contains a particular element.

- If you open up a simple Python shell (either using IDLE or the Terminal), you can test out this operator.

```
>>> 'a' in 'cat'
True
```

- It also works for longer strings, so you can try this:

```
>>> 'with' in 'Programming with Python is awesome'
True
>>> 'with Python' in 'Programming with Python is awesome'
True
```

- Watch out though, it's case sensitive!

```
>>> 'with python' in 'Programming with Python is awesome'
False
```

- If you don't care about case, then you can use the `lower` or `upper` string (A sequence of characters) method to convert the string (A sequence of characters) to a single case.

```
>>> 'with python' in 'Programming with Python is awesome'.lower()
True
```

- If you want to look only for complete words, you might encounter another problem.

```
>>> 'on' in 'Programming with Python is awesome'.lower()
True
```

- As you can see, the operator found the substring `on`, as it's the last two characters in the string (A sequence of characters) `Python`. If you only want to look for complete words, you can split the string (A sequence of characters) up first. This turns it into a list.

```
>>> 'Programming with Python is awesome'.lower().split()
['programming', 'with', 'python', 'is', 'awesome']
>>> 'on' in 'Programming with Python is awesome'.lower().split()
False
>>> 'python' in 'Programming with Python is awesome'.lower().split()
True
```

Don't worry if you're completely stuck - the hints below can help you out.

I need a hint

- Your trigger should remain the same: it should be the "LED" keyword.
- Within the `run` method, create a new variable called `command` that is equal to the `voice_command`, but converted to lower case.
- If the string `on` is in the `command` string, then the LED can turn on.
- If the string `off` is in the `command` string, then turn the LED off.

Here's a bit of code to get you started:

```
def run(self, voice_command):
        command = ## make it all lower case
        if "on" in command:
                self.say('Turning LED on')
                ## turn on the LED

        elif ## something here :
                self.say('Turning LED off')
                ## turn off the LED
```

Here's a little video showing you how to write the code:

What next ?

- Can you make the LED blink when the user asks it to blink?
- Can you use voice commands to control the speed of the blinking?
- What other components could you control? How about a servo?

Published by the Raspberry Pi Foundation – www.raspberrypi.org