**2018 – Voice controlled Velux with Snips**

Howdy fellas! Bored to run after your Velux remote controller because you can’t remember where you left it? Or maybe you simply would love to be able to use your voice to control them? You’re at the right place for that! Today we are going to review the entire process to implement Snips on your Velux.

This is the goal of this guide (yeah, just to tease you…): https://www.youtube.com/watch?v=ukkOLqcm2CY

**Disclaimer**

Just be aware that following this guide will void any warranty on your Velux remote. You are going to solder, cut, tape, glue, screw and **neither I nor Snips can be held responsible for any nonfunctional remote, damages or injuries**. This hack goes to a working state, without 3d print of a new casing so you are going to end up with an electronic component in the open! You will not be able to use the remote normally anymore but only through your voice or your computer keyboard. In other words, you know what you are doing here! I will gladly provide help on my free time and will update this guide for further addons (error handling, 3d printed casing, electronic board print).

**1. Material**

* Raspberry pi Zero W
* Raspberry power adapter
* SD card for Raspbian stretch
* A KLR 100 Velux remote. These were coming with your Velux, one per window, so if you didn’t get any, ask your retailer, you should have. They have now been replaced by the KLR 200 that we won’t use. If you don’t own any, try at a local carpenter, they usually stock the excess when mounting new windows at their customers. (ref img 1-1)
* Torx screwdriver
* 9 x REED Relay 0.5A 3 volts. I bought SIL03-1A72-71D from Digi-Key
* Bakelite test board or a breadboard board if you don’t want to solder (5.08mm steps, example: https://www.conrad.ch/fr/platine-dexperimentation-wr-rademacher-wr-typ-721-vk-c-721-bakelite-l-x-l-160-mm-x-100-mm-35-m-pas-508-mm-1-pcs-527645.html)
* 12 x GPIO male-female wires if you use a pre-soldered raspberry 0
* Some 0.25 wire
* Double sided tape
* Electronic soldering iron

**2. Hardware hack**

Waaiiiiit! Drop that screwdriver!! Before we start messing our Velux controller we must make sure it is connected to your installation as we are going to remove all the buttons, leaving you with no possibilities to interact manually with your windows until we have the program running! Reset your Velux controller and make sure:

* To add all your Velux components.
* **To not add any shortcut to P1 and P2!**
* **To turn off display sleeping by** by going to Menu => User settings => Display => Sleep mode => off. While you are there, turn on screen light as well, as the next guide will need that back light.

This tutorial covers windows and blinders, I don’t have any Velux lights unfortunately, but the process should be the same. Be sure to create the rooms on your remote and add them their respective components.

Ok, if you forget a few things, you’ll always be able to finish the remote installation with your keyboard at a later point. Let’s go!

* Turn your remote upside down
* Remove the battery cover and the batteries
* Unscrew the 2 torx screws (ref img 2-1)
* Use a blade to open the controller as it is clipped together (ref img 2-2)
* Remove the card from its plastic holder (ref img 2-3)
* The buttons are protected under a tiny plastic sheet, remove it. Turn the card around, there’s a lonely button, the reset button, that is covered as well, uncover it too (ref img 2-4)
* Unsolder the battery contacts, we won’t need them anymore and will power the card through our Raspberry. The top one is +, the bottom one is -. (ref img 2-5)
* Cut 16 wires of roughly 110mm in length, strip and tin them. I used black and red wires, old school, but red for + and black for –
* Cut a piece of bakelite test board of roughly 102x25mm. You’ll need a grid of at least 40x9 holes. Cut it in the height, so that you have 102mm connection lines
* We will power the relays from the pi gpios (3.3v) and one ground only. Check the scheme to cut the tracks. Schools teach you to cut the tracks with a drill. I use a Dremel, nicer and easier. Note that the 45° cut on the lower left corner is only for you to keep the track in the next images, you don’t need to cut it. Note the 0,0 coordinates on the top left corner. Imagine the holes as coordinates, the top left corner being 0,0 and the lower right hole being 9,40  
    
  IMAGE 2-6
* Solder the reed relays on the board. Beware, they have a polarity to respect! Make sure the pin 3 (+) is on the track that is cut intermittently and the pin 5 (-) is on the track that goes through the whole board  
    
  IMAGE 2-7
* Solder the gpio pins. Use the male side on the board and leave the female side for the raspberry  
    
  IMAGE 2-8
* Solder a red male-female wire on 7,3 and two black male-female wires on 1,40 and 7,40 (Blue spots on the image). These will go to the raspberry to get some 5v and 2 grounds.  
    
  IMAGE 2-9  
  IMAGE 2-10  
  IMAGE 2-11
* Let’s attack the Velux card now… We have 8 buttons to hack. We do not touch the P1 and P2 buttons as it’s useless to pilot them from the raspberry. Have a close look at one of the buttons. It is basically an outer ring and an inner contact. When you press the button, you technically connect the outer ring to the center. That’s why we made a little card covered with relays! Use the 16 red and black wires you’ve cut and prepared and solder the red ones to the inner part of the buttons, the black ones to the outer ring of the buttons. My advice? Prepare the buttons prior to soldering the wires to them with a drop of tin. Cut the stripped wires extremely short to avoid the cables touching both the center and the outer ring. Do that for the 8 buttons on the board. Solder the two remaining wires to the ex-battery sockets. On the following pictures I have protected the wires/buttons with some Bondic UV glue  
    
  IMAGE 2-12  
  IMAGE 2-13
* Breathe a little, take a break, do whatever you need, the next step is long and boring… All those wires now need to go to the relay board we made earlier… The relays center pin (3 and 5) are the coil pins, we are going to connect the buttons to the contact pins, 1 and 7. Polarity doesn’t matter here. First, I have numbered our buttons. This numbering will follow us to the end, as it is also used later when programming. Note that “10” is the button behind the card, the reset button. Also note that the image is an untouched Velux board, yours surely looks more like a messy hairy monster with red and black hairs.  
    
  IMAGE 2-14
* The numbers beside the relays on the next scheme represent the button numbers. Solder the red and black wires accordingly (black wires are drawn blue on scheme). Relay P pin 7 gets the red wire going to the Velux board top ex battery socket. Hole at 1,38 gets the black wire going to the Velux board bottom ex battery socket.  
    
  IMAGE 2-15  
  IMAGE 2-16  
  IMAGE 2-17
* Cut some double-sided tape of the same length as your relay board, glue it on the relay board, and glue the whole on the Velux board back!  
    
  IMAGE 2-18
* Take a break, breathe again, we’re through the hardware modifications.
* Let’s connect to the raspberry!
  + Connect the red wire from 3,7 to #4 (DC Power 5v)
  + Connect the black wire from 1,40 to #6 (Ground)
  + Connect the black wire from 7,40 to #34 (Ground)
  + Connect relay 10 (5,40) to GPIO #40 (GPIO21)
  + Connect relay 9 (5,36) to GPIO #37 (GPIO26)
  + Connect relay 8 (5,32) to GPIO #38 (GPIO20)
  + Connect relay 7 (5,28) to GPIO #35 (GPIO19)
  + Connect relay 5 (5,24) to GPIO #36 (GPIO16)
  + Connect relay 1 (5,20) to GPIO #33 (GPIO13)
  + Connect relay 2 (5,16) to GPIO #32 (GPIO12)
  + Connect relay 3 (5,12) to GPIO #31 (GPIO06)

**3. Programming**

We’ll do this in Python… I’m using Jetbrains PyCharm because it has everything at once, no need for Putty/Kitty, it auto uploads, has an ssh client etc etc.

I’ll assume that if you read and followed up to here, you have a running pi zero, meaning I won’t cover the Raspbian installation.

Make sure to install Snips following the wiki at <https://github.com/snipsco/snips-platform-documentation/wiki/1.-Setup-the-Snips-Voice-Platform>

And check if it is really running using “snips-watch -v”

I will not cover the entire programming part but simply share a **basic script** that you’ll be able to extend to your needs and explain how it works

* Go to your snips account and add my published bundle called “Velux” to your assistant and upload it to your raspberry by following this guide if not familiar: <https://github.com/snipsco/snips-platform-documentation/wiki/2.-Create-an-assistant-using-an-existing-bundle>
* The bundle has 4 intents:
  + openBlinders
    - **Slots**
      * place
      * percentage
  + closeBlinders
    - **Slots**
      * place
      * percentage
  + openVelux
    - **Slots**
      * place
      * percentage
      * duration
  + closeVelux
    - **Slots**
      * place
      * when
* **None of the slots are required.** If you don’t mention a place, our script will open all the windows / blinders
* Now we need to program the actions… It’s not hard, don’t be scared! Open the velux.py file with your favorite editor and head to line 62. You should find a dictionary called “\_COMMANDS”. This is nothing more than a listing of button presses you would make to execute an action using the remote! Say you had a normal remote, if you wanted to **fullOpen** the selected windows, you press **7** twice, rapidly.
* If you insert strings in the command array, it will set the **following** button presses delay. By default, between each button press you’ll have a 0.5 second wait time. Override it by using strings. Yes, “select….” Commands start with a ‘1.5’. That is because, if you try it yourself, pressing the “Locate” button, using keypad number 3 has a huge lag. We must make sure the remote has time to open the locate display before continuing and setting the delay to ‘0.25’.
* Let’s look at one example:  
  ‘selectBedroom’: [‘1.25’, 3, ‘0.25’, 5, 1, 1, 5, 5, 1]
  + Set the delay after next action to 1.25 seconds
  + Press button #3 (Locate) - *wait 1.25 seconds*
  + Set delay after next action to 0.25 seconds
  + Press button #5 (Arrow down) – wait 0.25 seconds
  + Press button #1 (Enter) – wait 0.25 seconds
  + Press button #1 (Enter) – wait 0.25 seconds
  + Press button #5 (Arrow down) – wait 0.25 seconds
  + Press button #5 (Arrow down) – wait 0.25 seconds
  + Press button #1 (Enter) – wait 0.25 seconds
* **This example will only work for me!** If you happen to have a room in second position with 2 Velux in it and blinders, then it will work for you too. Otherwise, you will need to create your own commands!
  + Create a new key in \_COMMANDS
  + The key naming starts with “select”, followed by the room name **as declared in the Snips intent slot “place”**, followed by either “Windows” or “Blinders.
  + Add a 1.25 seconds delay, a press on button 3 and a 0.25 seconds delay.
  + Add any key combo needed to select the product/s your command should trigger.
  + To make it clear and help understand those of you not really getting it:  
      
    IMAGE 3-1  
      
    This would translate to:   
    ‘selectBedroomWindows’: [‘1.25’, 3, ‘0.25’, 5, 1, 1, 5, 5, 1]
* If you want to operate your blinders, well, it’s the same! Blinders work just like windows, just add a ‘selectXXXBlinders’ with the needed combo!
* Once you have added all the commands you need, edit the \_INTENT\_ variables on lines 75 to 78 to match yours. Snips adds your username followed by a semi colon, match your username!
* Well, that’s about it! Upload the script to your raspberry, start it by typing “python velux.py”. As soon as the script starts, your controller should turn on and 12 seconds later a message asks you to type a button number. You don’t have to, I added this part for convenience, so that we can use our remote by using our keyboard. If you need to reset the controller, you can type reset and the controller will reboot and reset. Be aware that if you reset, you’ll have to add all your products back (or transfer from another remote, which is way easier).
* Take a deep breath and ask Snips to open your windows! Something as simple as “Snips, open my windows” should trigger them all to open to 100%

**4. The end, conclusion, enhancement etc**

That’s about it, we made it! If everything went fine, you are now able to control your Velux windows and your blinders with your voice. Of course, this is far from being perfect, but for the sake of this guide I decided to go only this far. You will soon notice that maybe it’s raining, and the remote will display an error. You have no feedback to your raspberry about that error and trying to control your windows again will cause problems. For this I have 2 solutions:

* Reboot the remote after each action, make sure your script has a timer to accept new commands. I made this, it works, it just adds a delay of around a minute between commands to your windows/blinders
* The solution I’m working on now is adding a very simple and cheap camera to read the screen every 2 seconds after an action is executed, compare the image to an image database of all the possible display states I made, and act accordingly by having the script press the needed buttons to go back to main screen. If you are interested in this solution, check back later or subscribe to my blog/twitter to be updated about it.

The script itself is also far from finished, I made something basic and simple so that it should be readable for even the python novices.

The relay board, even though working, is far from being professional. I have friends in the industry that proposed to print a real board with smaller components as well as a solderless contact board for the buttons. This is not yet realized.

I hope you enjoyed this guide and that you too realized your dream of having voice-controlled Velux! Subscribe to my channels and stay tuned for further hacks!

Laurent Chervet aka Psycho  
@Psychokiller188