HASduck

Installation and User Guide for RPI

Hudson Valley Digital Network

15 March 2021

v1.2

Table of Contents

Introduction

Welcome to HASduck!

> *Transmitter hunting (also known as T-hunting, fox hunting, bunny hunting, and bunny chasing), is an activity wherein participants use radio direction finding techniques to locate one or more radio transmitters hidden within a designated search area. This activity is most popular among amateur radio enthusiasts, and one organized sport variation is known as amateur radio direction finding.   --- Wikipedia*

Radio beacons used in transmitter Hunting are often simple low power transmitters sending a pre-recorded message (voice or CW) at pre-defined intervals. But what if we make the beacon smarter and give it the will and smarts not to be easily found. Smarts that include being aware of its environment and the ability to react to any changes in it. We would refer to these as Ducks with finding them called DuckHunts. This the HASduck concept was born.

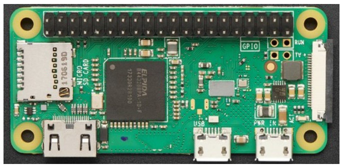
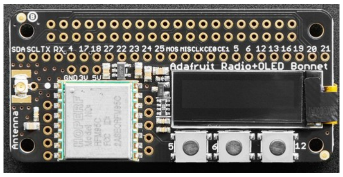
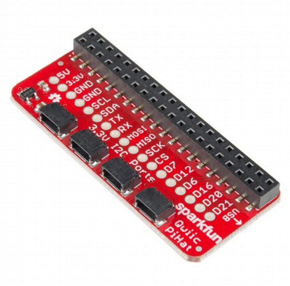
This guide will help you get started with HASduck beginning with required RPi hardware and OS installation. We will follow this with HASduck installation, configuration, and usage. All references to “RPi” in this guide are to the Raspberry Pi Zero WH hardware.

Hardware

HASduck consists of the following architecture components:

- Single board computer
- Display (OLED)
- Input (Pushbuttons)
- Network (WiFi)
- RF Module (LoRa)
- RF Antenna
- Power (2.5A)
  - Uses case requiring portable power find 1200maH delivers ~7 hours of RF at low power when sent at 5 second intervals.

Tested hardware includes the following:

Raspberry Pi Zero WH (Wireless with Headers) • You will need to solder the 20-pin header to the board.	
Adafruit LoRa Radio Bonnet with OLED – RFM95W @915Mhz • Requires an antenna connected via a male U.FL connector	
SanDisk Ultra 16GB RAM Class 10 MicroSD • RPi require quality microSD cards at least Class 10	
Power Source 5V @ 2.5A minimum • The hardware can be powered via USB to PC as part of setup but for operation it needs to be on a standalone power source.	
900 MHz Omnidirectional "stub" Antenna • Our antenna U.FL IPEX to SMA Connector • Connect the omnidirectional antenna to the Adafruit LoRa Radio Bonnet	
If you plan on connecting i2c sensors we recommend using the Sparkfun QwiiC Connect System adding the Sparkfun Qwiic HAT for RPI to your hardware stack.	

Software

Preparing the microSD

An overview of building HASduck starts with microSD card preparation;

- ○ From a computer download and run Raspberry Pi Imager
- ○ Within Raspberry Pi Imager, click on **CHOOSE OS** under **Operating System**, then **Raspberry Pi OS (Other)**, then **Raspberry Pi OS Lite (32-bit)**
- ○ Insert microSD card then click on **CHOOSE SD CARD** under **SD Card** then select SD card.
- ○ Click **WRITE** to begin formatting the SD card and installing the OS

Preparing the OS

With the OS image, we next need to mount the card and create/edit files to remotely connect to the RPi via WiFi or USB. If you plan on using a directly connected monitor and keyboard then you can jump to the Setup section.

Remote Connectivity

Existing Wireless Network

• Mount the SD card
• Change to the **boot** directory on the SD card
• Enable **ssh** access by creating a blank file named **ssh**
	`cd /media/sd-card/boot`
	`touch ssh`

• Within the boot directory, create a file called **wpa_supplicant.conf** and edit	`vi wpa_supplicant.conf`

• When you have opened the new file**,** add the configuration at right and save
• Be sure to replace **SSID** with your local wireless network SSID
	`country=US`
	`ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev`
	`update_config=1`
	`network={`
	`        ssid="MyWiFiNetwork"`
	`        psk="aVeryStrongPassword"`
	`        key_mgmt=WPA-PSK`
	`}`

• Change to your home directory, run **sync** as sudo, and unmount the SD card
• Remove the SD card
	`cd ~`
	`sudo sync`
	`umount /media/sd-card/root`
	`umount /media/sd-card/boot`

USB connection

If WiFi is not available, using zero-configuration networking services may be an option if you are running OSX or Windows with Apple BonJour Services installed. Linux users will need to consult their distribution documentation on adding USB as an IP interface and any configuration changes to the installed Avahi software.

• Change to the **boot** directory on the SD card
• Edit the **config.txt** file
	`cd /media/sd-card/boot`
	`vi config.txt`
	`Append the following line:`
	`dtoverlay=dwc2`
	`Then save the file.`

• While in the boot directory, edit the **cmdline.txt** file, replace a line, then save file.	`vi cmdline.txt`
	`Replace with the following all as one continuous line:`
	`dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait modules-load=dwc2,g_ether quiet init=/usr/lib/raspi-config/init_resize.sh`
	`Than save the file.`

• Remaining in the boot directory, enable **ssh** access by creating a blank file named **ssh**	`touch ssh`

Install

RPi Configuration

With the OS image created, we next insert the SD card into the Pi, boot the Pi and configure it.

- Insert SD card
- Connect RPI and Power on
- Log in with default pi:raspberry

If accessing via IP over USB, connect PC USB port to RPi USB Data port (next to mini-HDMI port)

ssh -l pi -o IdentitiesOnly=yes <yourpi>

If connecting via SSH for first time click **yes** to **accept fingerprint**

- Run Configuration tool

```
sudo raspi-config
```

- Navigate through each menu making selections as noted then exit tool

```
1. System Options
   S1 Wireless. LAN update (if using to connect)
   S2 Password.  (change password)
   S4 Hostname. Change hostname to your call + number [50-98]
      yourcall-50
   S5 Boot/Auto Login select B1
2. Display Option  (Nothing to do)
3. Interfacing options
   P2 SSH Enable
   P4 SPI Enable
   P5 I2C Enable
4. Performance Options
   P2 GPU Memory: Set to 16
5. Localization Options
   L1 Change Local to <your-local>.UTF-8
   L2 Change Timezone
   L4 WLAN Country
7. Advanced Options
   A1 Expand filesystem
8. Update
```

- Run **sync** as sudo, and **reboot** (ignore errors)

```
sudo sync
sudo reboot
```

HASduck Install and Initialize

With the RPI itself setup, we next install HASduck and setup its RF configuration (ID, LoRa mode, etc.)

- SSH into your Pi
- Run OS update and upgrade
- Run sync and reboot

```
sudo apt-get update
sudo apt-get -y upgrade
sudo sync
sudo reboot
```

- Log back in as pi and clone the HASduck repo
- Change directory into the cloned repo and run the install script

```
wget https://github.com/hudsonvalleydigitalnetwork/HASduck.git
./HASduck_install_fresh.sh
```

- Run **./HASviolet_config.py**
- Change **NOCALL** to your call or handle
- Change **SSID** to a number 50 – 99
- If needed increase **Transmit power** from 5 to as high as 20
- Change **beacon** as you see fit
- Save and exit

```
      -===- HASviolet Config Tool -===-

  - - - - - - - - - - - - - - - - - - - - -
  1     Change my Call / Handle  PURPLE
  2              Change my SSID  50
  3            Change my Beacon  QRZ QRZ QRZ
  4   Change dest Call / Handle  BEACON
  5            Change dest SSID  99
  - - - - - - - - - - - - - - - - - - - - -
 11                Change Radio  RFM9X
 12       Change Frequency (Hz)  911250000
 13       Change Transmit Power  10
 14                Change Modem  0
 15       Change Bandwidth (Hz)  125000
 16        Change Spread Factor  7
 17          Change Coding Rate  8
  - - - - - - - - - - - - - - - - - - - - -
 40      Show current HASduck.json
 41  Generate ModemRegister settings
  - - - - - - - - - - - - - - - - - - - - -
 51  Write changes to HASduck.json
  - - - - - - - - - - - - - - - - - - - - -
 99          About HASduck-config
  0                   Exit program

Select menu item:
```

- Run **./HASduck.sh install**

```
This installs HASduck to run on boot.

You will see the OLED light up with a menu.
```

Configure

At this point HASduck is ready to be configured as an active RF transmitting device. A default configuration is included for HASduck to behave as a simple beacon sending one group of commands starting with the **message “Quack Quack Quack”** every **ten seconds** for **one hour** using **N0CALL-99** or the call/handle you setup in **HASviolet_config.py**. We can change the “duck behavior” using the **HASduck_config.py** tool which writes to a JSON config file **HASduck.json**

Before we run the tool, we need to understand the five simple commands used in setting the ducks behavior;

- **message** is the text you want to send
- **interval** is how many seconds between re-transmission of message
- **trigger** is an event you can set that leads to an action. The command has two arguments of *type* and *value*. Types include;
  - **none**. Does nothing for the duration of the workload. Value 0 is assigned
  - **gps**. Change in location (duck moved) or some other gps value
  - **bluetooth**. Detection of any or specific BLE signal
  - **wifi**. Detection of any or specific WiFi signal
  - **custom** as written by yourself
- **action** is taken when the trigger has been tripped. Action has one argument which could be
  - **next** which runs a new group of commands
  - **reset** which re-runs the last set of commands
  - **quit** which stops beaconing altogether
- **duration** is time-out in seconds for the current group of commands

NOTE: Triggers remain under development as of this release and will be made available as plug-ins. For now trigger needs to remain none as type and 0 for value. Only one group of commands is supported.

- From **/home/pi/HASduck r**un **./HASduck_config.py**
- Select menu number to make your changes
- Save and exit
- For any changes to take effect, you must reboot HASduck

```
- - - - - HASduck Config Tool - - - - -

  1         Message  Quack Quack Quack
  2        Interval  10
  3         Trigger  none 0
  4          Action  reset
  5        Duration  3600

- - - - - - - - - - - - - - - -

  9     Quick Guide
 99   Write changes
  0       Exit Menu

Select option:
```

Usage

The left button (B1) is used to cycle through menu selections and the middle button (B2) is used to make your selection. The right button (B3) is a hard reset to get you out of any option selected. After boot up, the OLED displays the following menu options;

TRANSMIT
RECEIVE
DUCKHUNT
OPTIONS

Transmit

HASduck provides the ability to send out short canned messages when it is not setup as a Duck using the DUCKHUNT menu. It will transmit the canned message in the following format;

N0CALL-99>BEACON-99|<msg>

Where as **N0CALL-99** is the call/handle you configured with **HASviolet_config.py, BEACON-99** is static, and **<msg>** is one of the canned messages per menu options

- **TX QSL where <msg> is QSL**
- **TX QRZ** where <msg> is QRZ
- **TX BEACON** where <msg> is beacon as configured in HASviolet_config.py
- **RETURN** to previous menu

Receive

HASduck provides the ability to recieve when it is not setup as a Duck using the DUCKHUNT menu. In this menu you have the following options

- **RX ALL** to listen to all signals on the channel
- **RX BEACONS** to listen to all signals with BEACON-99 as destination on the channel
- **RX ME** to listen to all signals with your call/handle as destination on the channel
- **RETURN** to previous menu

Duckhunt

HASduck provides the ability to recieve when it is not setup as a Duck using the DUCKHUNT menu. In this menu you have the following options

- **SHOW DUCK** to see current **HASduck.json** settings
- **RUN DUCK** starts HASduck in DUCKHUNT mode. Let the games begin!
- **RETURN** to previous menu

Options

In this menu you have the following options

- **ABOUT DUCK** to show current HASduck version
- **RESTART** to reboot HASduck
- **QUIT** to stop running **HASduck.py**. Used when SSH/connected into HASduck for
- **RETURN** to previous menu

Development

Overview

a data communications application suite designed to be used on RF networks such as LoRa. Applications are written in Python 3.7.X using Visual Studio Code. Shell scripts (Bash) are used for creating the HASduck installation environment. The working directory for all applications is installed in ***/home/pi/HASduck***The application suite is developed in Python and includes the following core applications;

- **HASduck_config.py** to configure Duck behavior
- **HASviolet_config.py** to configure your station with your call sign/handle
- ***HASduck.py*** - Core program
- ***HASduck.sh*** - Install/removal script for HASduck.py to be run on startup
- ***HASduck.service*** **–** Service file used by HASduck.sh script
- **HASvioletRF.py** is a RF interface library used by all HASduck applications to support abstraction from the variety of RF libraries/modules expected to be support edover time.
- **HASvioletHID.py** is a HID library (OLED, Buttons, etc) used by all HASduck applications to support abstraction from the variety of modules/methods to be supported over time.
- **HASrf95.py** is a HOPE RFM95 library referenced by the **HASvioletRF** library
- **font5x8.bin** is a font file for OLEDs as used by the **HASvioletHID** library

Core and other applications have the dependency on the following config files stored in the **cfg/** directory;

- **hasDUCK.json** is a configuration file generated by **HASduck_config.py**
- **hasVIOLET.json** is a configuration file generated by **HASviolet_config.py**
- **hvdn-logo.xbm** is a bitmap graphic file used by the **HASvioletHID** library

HASduck runs as a daemon on startup using the following files;

- **HASduck.sh** is a script that can install, start, stop, and remove the HASduck services as a daemon on startup
- **HASduck.service** is a systemd file installed that starts HASduck.py
- **HASduck.py** is a service that runs on RPi bootup

Libraries

For ease of development by end-users, HASduck core applications in the previous release (Antigua) have been ported into two libraries to be used in your applications. The library names are **HASvioletRF.py** and **HASvioletHID.py**

HASvioletRF.py

All RF functions are performed through this library. It has dependencies on RF specific modules that currently include **HASrf95.py** for the HOPE RFM95 module currently used on the Adafruit Radio bonnet. Support for the ST126X module is planned and will be referenced via the HASvioletRF.py. Some of the variables available to user applications include the following

Variable	Type	Description
`self.radio`	string	`RF Modules used (RFM9X, Sx126X, etc)`
`self.modem`	string	`Modemstring as standardizedby Radiohead`
`self.frequency`	string	`Frequency in Hz`
`self.spreadfactor`	string	`LoRa Mode spreadfactor`
`self.codingrate4`	string	`LoRa Mode Coding Rate`
`self.bandwidth`	string	`LoRa Mode Bandwidth`
`self.spreadfactor`	string	`LoRa Mode spreadfactor`
`self.spreadfactor`	string	`LoRa Mode spreadfactor`
`self.txpwr`	integer	`Value from 5 to 23`
`self.mycall`	string	`Callsign or Handle (ie VIOLET)`
`self.myssid`	integer	`Recommended 50-99 (ie 50)`

HASvioletHID.py

All HID functions are performed through this library. This includes GPIO addressable buttons and OLED displays. This library has dependencies on the 128x32 SSD1306 display driver as available through the **adafruit_ssd1306** library. Some of the methods available to user applications include the following

Variable	Type	Description
`self.OLED`	method	`Frequency within 868 or 900 MHz bands in MHz (ie 911.25)`
`self.OLED.fill`	method	`Fill OLED screen`
`self.OLED.show`	method	`Show on OLED screen`
`self.btnLeft.value`	method	`Left Button position`
`self.btnMid.value`	method	`Middle Button position`
`self.btnRight.value`	method	`Right Button position`
`self.logo`	method	`Displays HVDN Logo on OLED`

Sensors

There are four interfaces that can be used to physically connect sensors and other devices to HASduck. They are:

- **SPI** which we reserve for the RF module given speed and full-duplex support
- **i2c** which is our preference for use with sensors connected using the Qwiic connect system.
- **microUSB** which we prefer for GPS devices
- **GPIO** as a last resort

The following sensor application “alpha code” is available in the **sensors/** directory.

- **HASviolet-atmos.py** for use with the Sparkfun BME280 Atmospheric sensor (Qwiic)
- **HASviolet-distance.py** for use with the Sparkfun Distance Sensor (Qwiic)
- **HASviolet-gps.py** for use with USB accessible GPS/GLONASS
- **HASviolet-sensors.py** for use with all three of the aforementioned sensors simultaneously

HASviolet-atmos.py

```
usage: HASviolet-atmos.py [-h] [-d DESTINATION]
HASviolet Atmos Sensor
  -h, --help            show this help message and optional arguments:
  -d DESTINATION, --destination DESTINATION
```

HASviolet-distance.py

```
usage: HASviolet-distance.py [-h] [-d DESTINATION]
HASviolet Distance Sensor
  -h, --help            show this help message and optional arguments:
  -d DESTINATION, --destination DESTINATION
```

HASviolet-gps.py

```
usage: HASviolet-gps.py [-h] [-d DESTINATION]
HASviolet GPS SensorHASviolet-distance.py
  -h, --help            show this help message and optional arguments:
  -d DESTINATION, --destination DESTINATION
```

HASviolet-sensors.py

```
usage: HASviolet-sensors.py [-h] [-d DESTINATION] [-a] [-f] [-g]
HASviolet Sensor TX
  -h, --help            show this help message and optional arguments:
  -d DESTINATION, --destination DESTINATION
  -a, --atmosphere      Atmosphere Sensor
  -f, --distance        Distance Sensor
  -f, --distance        Distance Sensor
```

Appendices