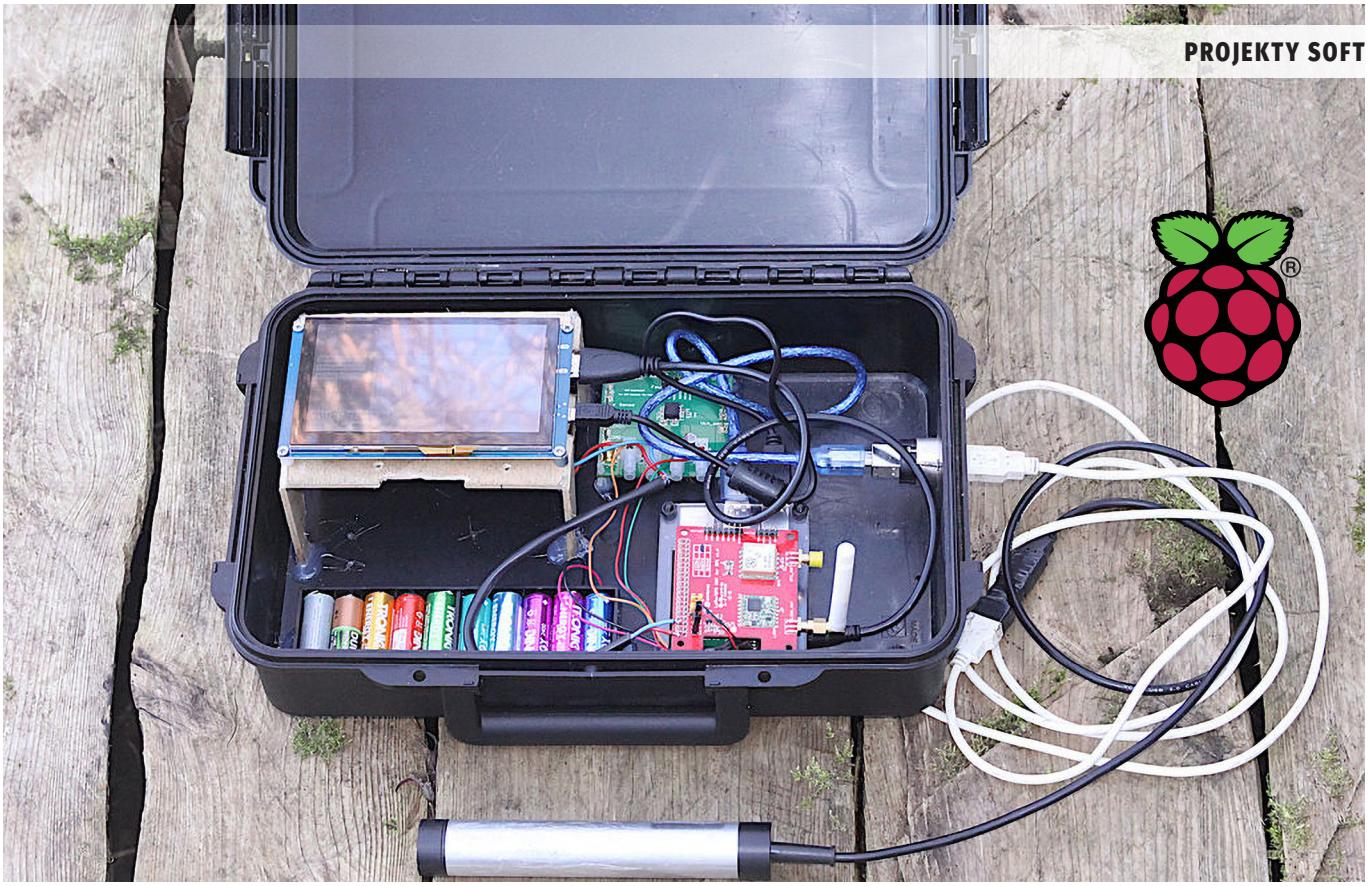


Inteligentny wykrywacz nietoperzy

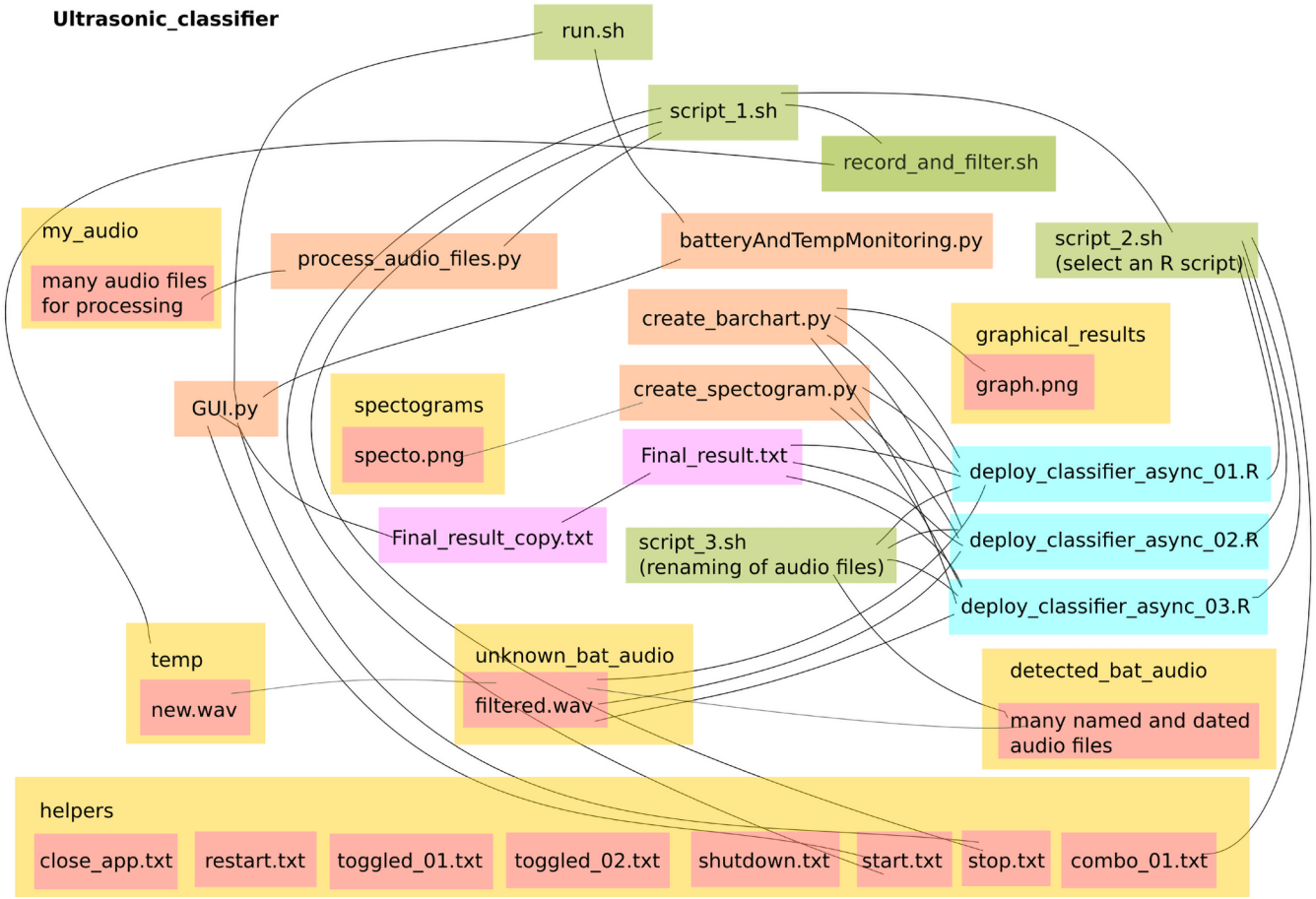
Nietoperze są w Polsce gatunkiem zagrożonym i podlegają one ścisłej ochronie. Populacja tych fascynujących ssaków jest cały czas monitorowana. Zwierzęta te wykorzystują echolokalizację – rodzaj „widzenia za pomocą dźwięku" – do rozpoznawania okolicy w ciemnościach. Z uwagi na fakt, iż każdy gatunek nietoperza wydaje trochę inne dźwięki, nagrywanie i analizowanie sygnałów echolokacyjnych to świetny sposób na monitorowanie i określanie gatunku nietoperza, jaki przelatuje nad naszymi głowami.

Nietoperze korzystają z systemu echolokalizacji. Emitują dźwięk o wysokich częstotliwościach, który odbija się od przeszkód, owadów etc. i wraca do ich czułych uszu. Pozwala to na „widzenie" w ciemności. Charakter emitowanych dźwięków pozwala zidentyfikować gatunek nietoperza, nawet, jeżeli znajduje się on dosyć daleko od nas. Jest to bardzo istotna informacja dla biologów monitorujących ilość nietoperzy na danym terenie. Analiza ta jest szczególnie przydatna, gdyż może być zautomatyzowana z wykorzystaniem opisanego poniżej prostego klasyfikatora. Dzięki temu, wykorzystując przenośny system oparty o komputer jednopłytkowy Raspberry Pi i dedykowane oprogramowanie możliwe jest wykrywanie i klasyfikowanie nietoperzy w terenie.

Prezentowane urządzenie wykrywające nietoperze może być zasilane z baterii 12 V i może być umieszczone w terenie na kilka dni lub nawet tygodni. Dane są przesyłane, co jakiś czas za pośrednictwem łącza radiowego LoRa. Zasada działania układu jest bardzo prosta – układ nagrywa 30 sekundowe próbki dźwięku i dokonuje jego analizy za pomocą systemu AI. Jeśli system wykryje w sygnale dźwięki charakterystyczne dla nietoperza to zmienia nazwę pliku na zidentyfikowany gatunek, dodaje do niego współczynnik określający pewność identyfikacji i datę. Pozostałe próbki audio są kasowane, by zaoszczędzić miejsce na dysku i zredukować obciążenie interfejsu komunikacyjnego.

Podstawowe cechy urządzenia:

- Mikrofon z pełnym widmem pomiarowym, zdolny do nagrywania dźwięku monofonicznego w jakości 384 kbps,
- Rezultaty mogą być wyświetlane w czasie rzeczywistym na dołączonym ekranie LCD, w postaci informacji o zidentyfikowanym gatunku lub jako widmo sygnału,
- Zasilanie z akumulatora 12 V lub dowolnego innego źródła zasilania o napięciu od 6 V do 16 V,
- Oprogramowanie zoptymalizowane pod kątem minimalizacji poboru prądu i wydajności obliczeń, dzięki wykorzystaniu klasyfikacji asynchronicznej,
- Średni czas pracy na 10 ogniwach 1,2 V NiMH typu AA to około 5 godzin,
- System zachowuje dane, nawet, jeżeli pewność detekcji wynosi 1%, ale ustawić można limit danych, po przekroczeniu, którego najgorsze pomiary będą kasowane,
- Możliwość przetwarzania wcześniej zebranych danych i danych z innych źródeł,
- Wykorzystanie w pełni otwartego oprogramowania,
- Możliwość uzupełniania modelu – trenowanie systemu uczenia maszynowego do wykrywania innych gatunków nietoperzy np. z innych obszarów geograficznych.



Rysunek 1. Schemat architektury oprogramowania systemu do wykrywania i identyfikacji nietoperzy

Wyzwania i problemy projektowe

Podczas realizacji projektu jego autor natknął się na szereg problemów związanych z koniecznością dobrania odpowiedniego rozwiązania technicznego do urządzenia. Pierwszym z nich był wybór odpowiedniego oprogramowania. Początkowo planował użyć pakietu do klasyfikacji muzyki „PyAudioAnalysis", który dawał możliwość wykorzystania algorytmów Random Forest, jak i rozpoznawania algorytmami uczenia maszynowego przy użyciu Tensorflow. Oba systemy działały nieźle, ale nie dawały zadowalających rezultatów. Po dalszych poszukiwaniach autor znalazł i zaimplementował pakiet napisany w R, który po zaledwie kilku godzinach optymalizacji, zapewnił bardzo dobre wyniki.

Drugim problemem było znalezienie danych audio dla nietoperzy dla kraju, w którym mieszka autor. Okazało się, że nie jest to takie proste, gdyż dane te są albo w ogóle niedostępne albo dostępne tylko dla wybranej grupy osób. Spowodowało to konieczność samodzielnego wyruszenia na audio-łowy i odnalezienie oraz nagranie kilku okolicznych gatunków nietoperzy. Autorowi udało się pozyskać próbki dźwiękowe od 6 gatunków tych zwierząt.

Jedynym problemem sprzętowym w systemie, było znalezienie przyzwoitego zasilacza. Musiał składać się z odpowiedniej baterii i przetwornicy zapewniającej napięcie 5 V. Jednocześnie musiał być wyposażony w możliwość pomiaru napięcia ogniwa i napięcia zasilania. Co zadziwiające, okazało się, że nie ma na rynku takich modułów dla Raspberry Pi. Utrudnieniem był dodatkowy wymóg – by częstotliwość pracy przetwornicy wynosiła powyżej 384 kHz. Dlatego w systemie zastosowano płytkę ewaluacyjną przetwornicy monolitycznej i osobny moduł z przetwornikami analogowymi podłączonymi do Raspberry Pi.

Oprogramowanie systemu także przyniosło szereg problemów. Projektowanie ogólnej architektury aplikacji było dosyć złożone. Na **rysunku 1** pokazano schemat całego software systemu. Za interfejs użytkownika odpowiada GTK 3, a reszta opiera się na różnych skryptach napisanych w Bashu i Pythonie do interakcji z głównym skryptem napisanym w języku r. Podczas gdy sam Python czy Bash są bardzo dobrze udokumentowane, to rozwiązywanie bardziej szczegółowych problemów z GTK 3 było bardzo trudne z uwagi na średniej jakości dokumentację jak i niewielkie wsparcie społeczności. Jak pisze sam autor „W porównaniu ze zwykłym programowaniem w Pythonie, GTK nie było przyjemnym doświadczeniem, chociaż bardzo satysfakcjonujące jest obserwowanie aplikacji w akcji". Natomiast nauka języka R nie była problemem, ponieważ jest to bardzo ogólny i dobrze opisany język.

Dużym problemem była równoległość prowadzenia obliczeń w oprogramowaniu. Niektóre części systemu, w szczególności moduł do nagrywania dźwięku na żywo, muszą być wykonywane płynnie. Zostało to osiągnięte w Bashu przy użyciu niewiarygodnie prostej składni – znak & (ampersand) i polecenie wait. Wszystko to odbywa się w dwóch bardzo schludnych wierszach kodu:

```
arecord -f S16 -r 384000 -d ${chunk_time} -c
-device=plughw:r/home/tegwyn/ultrasonic_classifier/
temp/new.wav &
wait
```

Wybór środowiska Bash do nagrywania fragmentów audio był dość oczywisty, ze względu na łatwość korzystania z bibliotek Alsa i zdolność do nagrywania w jakości 384 kbps. Bash pozwolił również na ścisłą kontrolę kolejności uruchamiania poszczególnych modułów, jako że niektóre bloki kody muszą być uruchamiane liniowo – działać jeden po drugim. Jest to najbardziej oczywiste w przypadku wdrażania modeli Random Forest, ponieważ trzeba je ładować do pamięci tylko raz na sesję, zamiast ładować za każdym razem, gdy wymagana jest klasyfikacja. Organizacja całego stosu była finalnie dosyć skomplikowana, ale dzięki dokumentacji każdego skryptu wszystko działa w zoptymalizowany sposób. Poszczególne skrypty komunikują się między sobą poprzez odpytywanie różnych plików tekstowych w katalogu „helpers", które bardzo często nie zawierają nawet żadnej zawartości.

Potrzebne moduły i elementy

Do zestawienia prezentowanego urządzenia potrzebne będą:

- Raspberry Pi 4 z 4 GB pamięci RAM,
- Obudowa z wentylatorem. Dobrze jest także nakleić radiatory na układach Raspberry Pi,
- Moduł ADC Pi do monitorowania napięcia baterii i napięć zasilających,
- Płytka ewaluacyjna EVM3683-7-QN-01A do dostarczania 5 V,
- Moduł HAT Dragino LoRa GPS,
- Sensor temperatury i wilgotności Adafruit Si7021,
- Wyświetlacz TFT LCD (800×480) o przekątnej 5” z interfejsem dotykowym,
- Bateria 12 V lub inne podobne źródło zasilania,
- Mikrofon USB UltraMic 384 lub inny o szerokim pasmie (zapewniający jakość 384 kbps) obejmującym ultradźwięki,
- Oporniki 270 kΩ, 3,3 kΩ oraz 48,1 kΩ,
- Wodoodporna obudowa na system,
- Pakiet oprogramowania do środowiska R WavX bioacoustics do analizy i ekstrakcji danych akustycznych nietoperzy,
- Oprogramowanie do klasyfikacji Random Forest.

Budowa systemu

Montaż urządzenia zrealizować można w kilku prostych krokach:

1. Podłącz moduł z ADC do Pi. Mogą być wymagane dodatkowe kable przedłużające, by podłączyć mierzone sygnały,
2. Zainstaluj wentylator – Raspberry Pi 4 musi mieć wentylator. Powinien on być podłączony do jednego z pinów zasilania 5 V,
3. Znajdź na płytce ewaluacyjnej R10 i zastąp go rezystorem o rezystancji 270 Ω i niskiej tolerancji, najlepiej ±0,1%. Sprawdź, czy na wyjściu dostępne jest teraz napięcie 5,0 V. Na wejście modułu podłączyć można napięcie do 16 V,
4. Napięcie z akumulatora 12 V podłączamy do pinu 1 w module ADC poprzez opornik 48,1 kΩ,
5. Napięcie z wyjścia zasilacza 5 V podłączamy do pinu 2 modułu ADC poprzez rezystor 3,3 kΩ oraz bezpośrednio do styków 5 V na Raspberry Pi,
6. Do komputera podłączamy interfejsy USB i HDMI z ekranu dotykowego,
7. Po załączeniu włącznika na płytce zasilacza system uruchomi się i będzie gotowy do instalacji oprogramowania.

Oprogramowanie

Aby zainstalować całe niezbędne oprogramowanie na Raspberry Pi, musimy postępować zgodnie z poniższą instrukcją. Z uwagi na rozległość oprogramowania, nie będziemy prezentować tutaj kodu programu ani jego fragmentów – wszystkie pliki zaleźć można w repozytorium na GitHubie autora.

1. Na karcie Flash micro SD o pojemności 128 GB należy wgrać najnowszą wersje Raspbiana z wykorzystaniem dowolnego programu do nagrywania obrazów, jaki używamy,
2. Po uruchomieniu Raspberry Pi z nowym systemem, musimy utworzyć nowego użytkownika o loginie *tegwyn* (zamiast domyślnego loginu *pi*). Aby to zrobić wchodzimy w terminal i wpisujemy następujące komendy:

```
sudo adduser tegwyn
sudo adduser tegwyn sudo
echo ‘tegwyn ALL=(ALL) NOPASSWD: ALL’ | sudo tee /
etc/sudoers.d/010_tegwyn-nopasswd
sudo pkill -u pi
```

3. Logujemy się, jako tegwyn i pobieramy z repozytorium wszystkie niezbędne pliki. W tym celu w linii komend wpisujemy:

```
git clone https://github.com/paddygoat/ultrasonic_
classifier.git
```

4. Odnajdujemy plik u*ltrasonic_classifier_dependancies_install_nano.sh* w folderze *home/ultrasonic_classifier* i otwieramy go z pomocą dowolnego, używanego przez nas edytora tekstowego. W pliku znajduje się link do bzip2-1.0.6, który musimy manualnie pobrać i zainstalować przed instalacją innych zależności,
5. Uruchamiamy cały skrypt do instalacji zależności. W tym celu w terminalu wpisujemy:

```
cd /home/tegwyn/ultrasonic_classifier/
chmod 775 ultrasonic_classifier_dependancies_
install.sh
bash ultrasonic_classifier_dependancies_install_
nano.sh
```

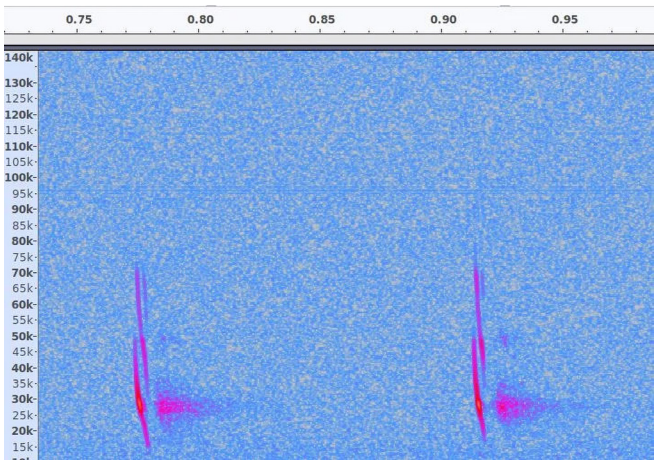
6. Odnajdujemy plik *run.sh* w folderze *home/ultrasonic_classifier* i edytujemy tam hasło w linii 156. Ponadto usuwamy z pliku linię 6 (`sudo -S jetson_clocks`), jako że jest ona dedykowana jedynie do pracy z system na platformie Nvidia Nano,
7. Na pulpicie powinna pojawić się ikona nietoperza. Wystarczy w nią kliknąć, aby uruchomić program. Widok okna programu pokazano na **rysunku 2**.

Rysunek 2. Okno programu do nagrywania i klasyfikacji dźwięków nietoperzy

Po uruchomieniu programu przez ikonę nietoperza na pulpicie można już zacząć korzystać z programu. Wybieramy opcję „Record and Classify” i naciskamy przycisk opisany „Graphical reporting”. Aby przetestować działanie systemu możemy zadzwonić metalowymi kluczami do mikrofonu – system po nagraniu i przeanalizowaniu tego dźwięku powinien go rozpoznać i wyświetlić w kolumnie, jako „HOUSE_KEYS”. Odświeżanie pomiarów odbywa się, co około 30 sekund. Jeśli chcemy, program pozwala także na oglądanie widm dźwięku w funkcji czasu. Przykład takiego widma pokazano na **rysunku 3**.

Nikodem Czechowski, EP

Źródło: ***http://bit.ly/2Vka5MA***



Rysunek 3. Widmo nagranego dźwięku – widoczne są miejsca, w których zarejestrowano dźwięk