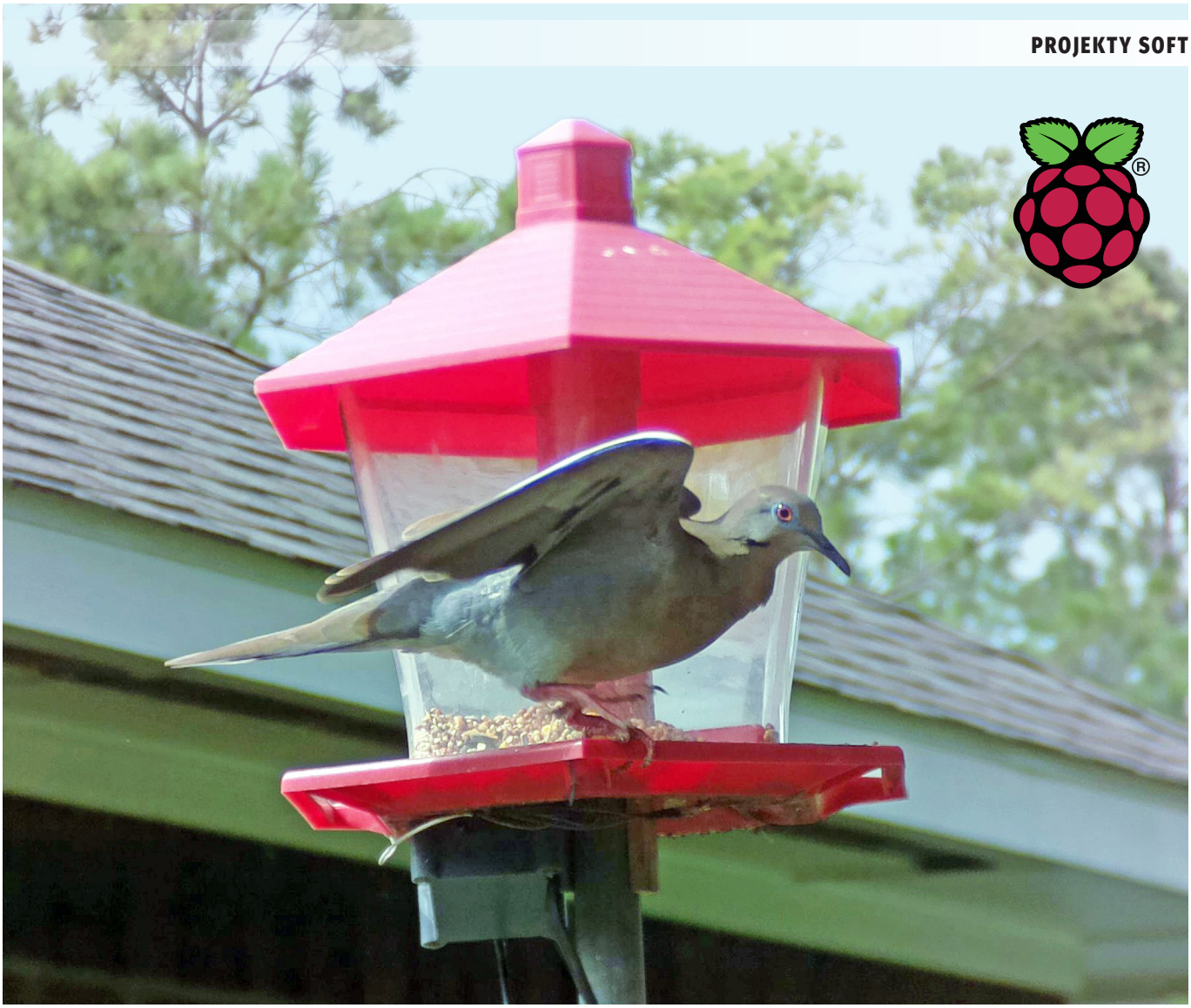


Cyfrowo monitorowany karmnik dla ptaków

W zimie na naszych oknach, balkonach i w ogrodach pojawiają się karmniki dla ptaków, którym chcemy pomóc przetrwać zimę. Jest to doskonała okazja do obserwacji tych niezwykłych zwierząt i do bliższego przyjrzenia się ich zwyczajom.

Zima jest doskonałym momentem do obserwacji ptaków, jednak nie zawsze możemy umieścić karmnik w miejscu zapewniającym nam dogodne do tego warunki. Właśnie to było inspiracją dla Stephena Kirbiego, który zbudował karmnik dla ptaków z elektronicznym systemem monitorowania, który jest zintegrowany z automatycznym aparatem fotograficznym oraz systemem do zbierania danych. Urządzenie ma na na celu monitorowanie, fotografowanie i rejestrowanie liczby i czasu pobytu ptaków, odwiedzających karmnik. Do budowy użyto kilku modułów Raspberry Pi, pojemnościowy czujnik dotykowy (Adafruit CAP1188) oraz moduł kamery dedykowanej do komputerów jednopłytkowych Raspberry Pi.

Potrzebne elementy

Do zestawienia prezentowanego tutaj systemu potrzebować będziemy między innymi (pominięto część drobnych elementów montażowych itp.):

- Raspberry Pi Zero W (do pojemnościowego sensora obecności ptaka),
- Raspberry Pi 3 B+ (jako serwer MQTT),
- dowolne Raspberry Pi z modułem kamery (opcjonalne, jeśli chcemy robić zdjęcia),
- wodoodporne obudowy na komputery jednopłytkowe i sensor pojemności,
- moduł sensora pojemnościowego Adafruit CAP1188,
- samoprzylepna taśma miedziana,
- zasilacze 5 V DC dla każdego komputera jednopłytkowego,
- złącza, kable, dławnice do obudów etc.,
- karmnik dla ptaków (oczywiście).

Działanie systemu

System monitorowania został zaprojektowany do liczenia, mierzenia czasu, rejestrowania i fotografowania ptaków jedzących z naszego

karmnika. Poprzednia wersja tej konstrukcji korzystała z Arduino Yun i zapisywała dane w arkuszu kalkulacyjnym na dysku Google. Nowa wersja, opisana tutaj, używa wiele komunikujących się ze sobą komputerów jednopłytkowych Raspberry Pi, protokół MQTT i lokalny magazyn zdjęć i danych.

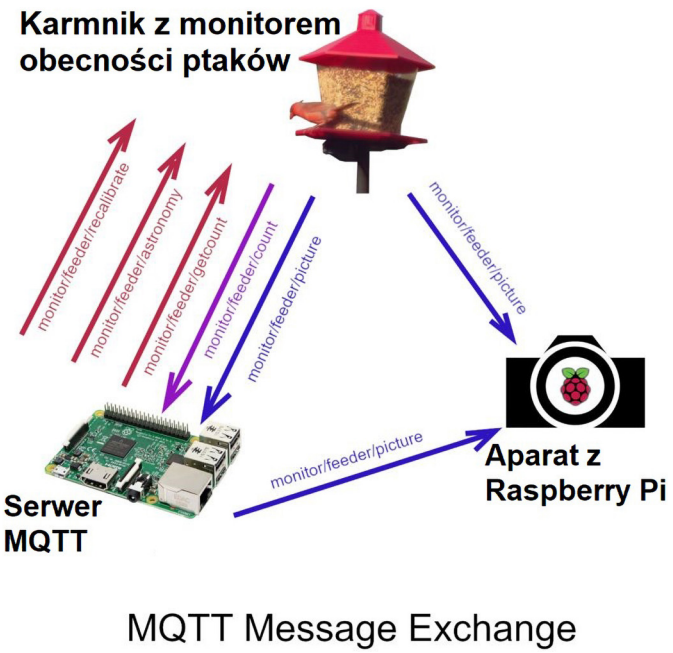
Karmnik dla ptaków jest wyposażony w Raspberry Pi Zero W i pojemnościowy czujnik dotykowy. Wszelkie ptaki siadające na monitorowanych żerdziach karmnika aktywują czujnik, który uruchamia z kolei licznik czasu, aby określić czas trwania każdego zdarzenia. Gdy tylko czujnik dotykowy zostanie aktywowany, komunikat MQTT *monitor/feeder/picture* zostanie opublikowany przez monitor karmnika. Wiadomość ta informuje aparat z Raspberry Pi o konieczności zrobienia zdjęcia. Jeśli serwer MQTT opublikuje komunikat *monitor/feeder/getcount*, monitor karmnika odpowie komunikatem MQTT *monitor/feeder/count*, który serwer zapisze w bazie danych. Dokładny przepływ komunikatów pomiędzy poszczególnymi blokami urządzenia pokazany został na **rysunku 1**.

Serwer MQTT realizuje w systemie kilka zadań. Żąda i przechowuje dane z monitora karmnika i kontroluje jego działanie. Aktywuje monitor o świcie i wyłącza o zmierzchu. Kontroluje również interwał czasowy żądania danych, a także monitoruje bieżące warunki pogodowe za pośrednictwem API DarkSky. Pogoda jest monitorowana z kilku powodów. Przede wszystkim opady mogą wpływać na czujniki – jeśli tak się stanie, czujniki są kalibrowane. Drugim powodem jest monitorowanie i rejestrowanie warunków pogodowych pod kątem korelacji ich z danymi dotyczącymi liczby ptaków.

Kamera Raspberry Pi to moduł RPi z dedykowanym modułem kamery. Oprogramowanie aparatu użyte w tym projekcie nie działa z kamerami na USB. Moduł ten jest wyposażony w interfejs WiFi i obsługuje oprogramowanie klienta MQTT. Subskrybuje wiadomości MQTT *monitor/feeder/picture* i za każdym razem, gdy taki komunikat otrzyma, robi zdjęcie. Zdjęcia są przechowywane na karcie SD kamery, ale zarządzane zdalnie.

Detektor obecności ptaków w karmniku

Komputer Raspberry Pi Zero W (RPi) i moduł CAP1188 są połączone ze sobą za pomocą interfejsu I²C. Na rynku dostępne są inne pojemnościowe czujniki dotykowe z jednym, pięcioma lub ośmioma czujnikami. Do tego projektu wybrano układ z ośmioma wejściami,



Rysunek 1. Przepływ komunikatów pomiędzy poszczególnymi blokami urządzenia

ponieważ karmnik, jaki będzie monitorowany, ma sześć boków, a każdy z nich jest instalowany niezależnie. Sposób podłączenia modułu CAP1188 do Raspberry Pi i sensorów zapisano w **tabeli 1**.

Zasilanie RPi jest dostarczane zewnętrznie (5 V podłączone do pinów 1 i 14) z zasilacza sieciowego 5 V DC. Układ nie pobiera zbyt wiele prądu, więc wydajność prądowa zasilacza nie jest krytyczna. Jednakże, jeżeli planujemy doprowadzenie napięcia zasilającego (5 V) z oddalonego zasilacza, dobrze jest zastosować przewody o odpowiednio dużym przekroju, by ich rezystancja nie powodowała nadmiernych spadków napięcia. Moduł ten można również zasilać z baterii lub powerbanku, jednak utrudnia to korzystanie z systemu, gdyż wymaga okresowej wymiany baterii.

RPi Zero i CAP1188 mogą znajdować się w jednej obudowie, koniecznie odpornej na warunki atmosferyczne, ale można też umieścić je osobno. Istotne jest, aby obudowa z CAP1188 znajdowała się blisko grzęd, na których naklejono miedzianą taśmę dla sensorów pojemnościowych.

Konfiguracja oprogramowania

Po zainstalowaniu systemu operacyjnego (Raspbiana np. poprzez NOOBSa) należy zalogować się do systemu, celem konfiguracji i instalacji potrzebnego oprogramowania. W pierwszej kolejności aktualizujemy system:

```
sudo apt-get update
sudo apt-get upgrade
```

Następnie uruchamiamy narzędzie do konfiguracji komputera:

```
sudo raspi-config
```

gdzie włączamy SPI, I²C oraz (jeżeli chcemy konfigurować komputer dalej zdalnie) SSH.

Instalację potrzebnych pakietów rozpoczynamy od instalacji pip-a:

```
sudo apt-get install python3-pip
```

Następnie instalujemy Adafruit CircuitPython:

```
sudo pip3 install --upgrade setuptools
```

Sprawdźmy, czy urządzenia interfejsów SPI oraz I²C są widoczne w systemie:

```
ls /dev/i2c* /dev/spi
```

Odpowiedź powinna wyglądać następująco:

```
/dev/i2c-1 /dev/spidev0.0 /dev/spidev0.1
```

Następnie instalujemy pakiety GPIO i Adafruit blinka:

```
pip3 install RPI.GPIO
pip3 install adafruit-blinka
```

W dalszej kolejności instalujemy oprogramowanie (bibliotekę w Pythonie) do obsługi modułu Adafruit CAP1188:

```
pip3 zainstaluj adafruit-circuitpython-cap1188
```

Finalnie, zainstalować możemy narzędzia do konfiguracji i diagnozowania interfejsu I²C:

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

Za pomocą powyższego narzędzia możemy sprawdzić, jakie urządzenia podłączone są do tego interfejsu:

```
i2cdetect -y 1
```

Jeśli CAP1188 jest podłączony, to w tabeli, która zostanie wydrukowana na ekranie, pod adresem układu zobaczymy, zamiast kreski powtórzony adres. Układ ten, zależnie od konfiguracji, powinien zgłosić się pod adresem 0x28 lub 0x29. Kolejnym krokiem jest zainstalowanie pakietu *mosquito*, wraz z jego klientem oraz instalacja MQTT:

```
sudo apt-get install mosquitto mosquitto-client
python-mosquitto
sudo pip3 install paho-mqtt
```

Adafruit oferuję doskonały tutorial dotyczący konfiguracji MQTT na Raspberry Pi (***http://bit.ly/2Q1XLMJ***). Postępujmy zgodnie z nim, by poprawnie skonfigurować i uruchomić MQTT na tym komputerze. Po zainstalowaniu i uruchomieniu MQTT można zainstalować na komputerze gita i z jego pomocą pobrać oprogramowanie sterujące systemem:

Tabela 1. Opis połączeń modułu CAPO1188 z Raspberry Pi i sensorami

Moduł CAP1188	Raspberry Pi/sensory	Opis sygnału
SDA	Pin 3	Pin danych interfejsu I²C
SCK	Pin 5	Pin zegara interfejsu I²C
VIN	Pin 1	Zasilanie (3,3 V)
GND	Pin 9	Masa
C1...C8	Wyprowadzone do gniazda, do którego podłączane są sensory	Sygnały sensorów
AD	Napięcie 3,3 V.	Adres interfejsu I²C CAP1188. Dla stanu wysokiego równy jest 0x28

```
cd ~
sudo apt-get install git
git clone "https://github.com/sbkirby/RPi_bird_
feeder_monitor.git"
```

Program potrzebuje folderu do przetrzymywania logów ze swojej pracy, w związku z tym musimy go utworzyć w głównym folderze komputera:

```
cd ~
mkdir logs
```

Teraz, po podłączeniu czujnika i zainstalowaniu wszystkich potrzebnych komponentów i uruchomieniu serwera MQTT, przystąpić można do testów działania systemu. W pierwszej kolejności edytujemy plik konfiguracyjnych *config.json*:

```
cd RPi_bird_feeder_monitor
sudo nano config.json
```

W pliku tym zastępujemy wartości dla „OIP_HOST", „MQTT_USER", „MQTT_PW" i „MQTT_PORT", wartościami dla naszej konfiguracji lokalnej. Zapisujemy zmiany w tym pliku i wychodzimy. Będąc nadal w katalogu */home/pi/RPi_bird_feeder_monitor*. Edytujemy jeszcze kolejny plik:

```
nano launcher.sh
```

Dodajemy następujący tekst w pliku *launcher.sh*:

```
#!/bin/sh
# launcher.sh
cd /
cd home/pi/RPi_bird_feeder_monitor
sudo python3 feeder_mqtt_client.py
cd /
```

Skrypt ten przejdzie do katalogu domowego, następnie do katalogu z programem do monitorowania, gdzie uruchomi skrypt Python, a następnie z powrotem przejdzie do katalogu domowego. Możemy teraz wyjść i zapisać skrypt *launcher.sh* i nadać mu odpowiednie uprawnienia (aby był wykonywalny). Po nadaniu mu uprawnień, możemy uruchomić go z użyciem sh.

```
chmod 755 launcher.sh
sh launcher.sh
```

Następnie musimy edytować *crontab* (menedżer zadań w Linuxie), aby uruchomić skrypt podczas uruchamiania komputera.

```
sudo crontab -e
```

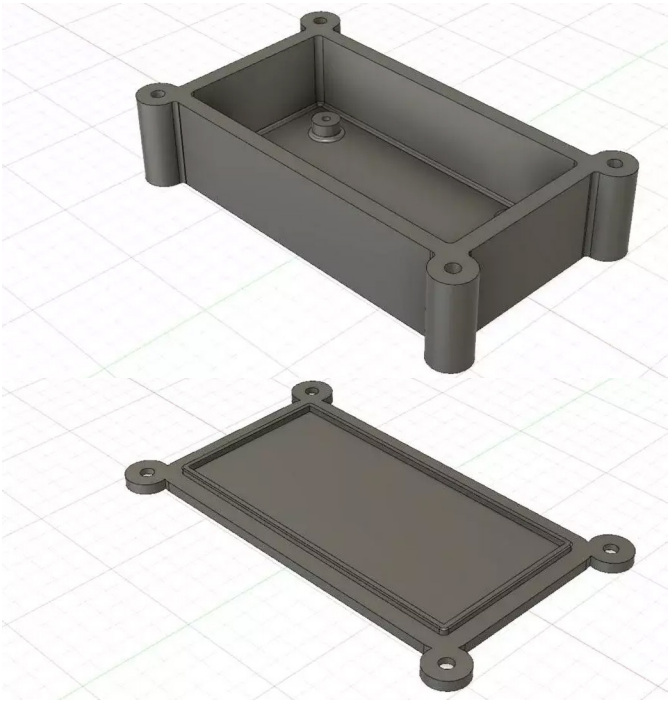
Spowoduje to wyświetlenie okna *crontaba*, gdzie na końcu pliku wprowadzamy następujący wiersz:

```
@reboot sh /home/pi/RPi_bird_feeder_monitor/
launcher.sh>/home/pi/logs/cronlog 2>&1
```

Po dopisaniu tej linii wystarczy normalnie zapisać plik i wyjść. Teraz można ponownie uruchomić Raspberry Pi. *Crontab* przy rozruchu powinien uruchomić skrypt, który z kolei uruchomi Pythonowy skrypt *feeder_mqtt_client.py*. Status skryptu można sprawdzić w plikach dziennika znajdujących się w folderze */logs*.

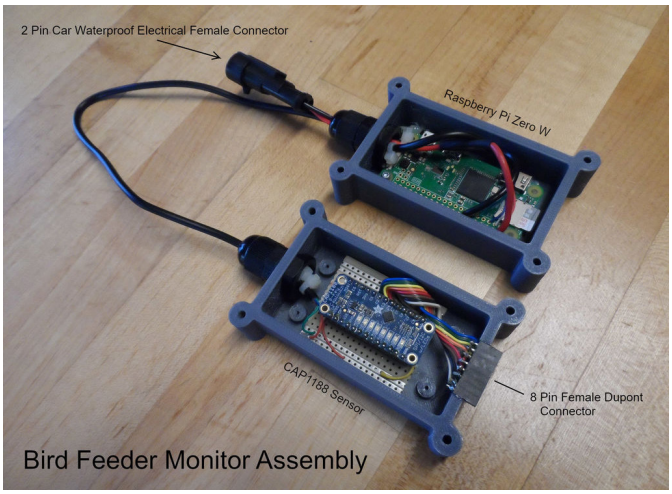
Obudowa i montaż urządzenia

Na stronie projektu znaleźć można pliki STL z elementami obudowy do druku 3D. Są one zupełnie opcjonalne – system ten można umieścić w dowolnej obudowie i tak długo, jak będzie ona wodoodporna, nie ma problemu z jej stosowaniem. Na **rysunku 2** zaprezentowano przykładową obudowę do druku 3D, jaką zaprojektował i użył autor projektu. Mieści się w niej Raspberry Pi Zero W oraz płytka uniwersalna z modułem CAP1188.



Rysunek 2. Obudowa do druku 3D, jaką zaprojektował i użył autor projektu

Obudowa pokazana w projekcie jest dwuczęściowa – można ją zmontować jako jedną obudowę na moduł Raspberry Pi i sensor pojemnościowy lub dwie osobne. W tym drugim przypadku, trzeba w niej wykonać otwór o średnicy około 12...15 mm na dławnice dla kabli, łączących Raspberry Pi i moduł pojemnościowy (**fotografia 3**).



Fotografia 3. Elementy urządzenia umieszczone w dwóch obudowach

Przez jedną z dławnic (w obudowie RPi) dodatkowo przechodzić będzie kabel do zasilania komputera. W obudowie z sensorem pojemnościowym zainstalowane jest 8-pinowe złącze Dupont dla sensorów pojemnościowych.

Przed zamknięciem obudowy na krawędzie nakładane są silikonowe uszczelnienia krawędzi, Dodatkowo, silikon dodać można na tyle złącz i przy dławnicach, aby zapewnić dodatkowe zabezpieczenie przed dostaniem się wody do układu np. podczas opadów deszczu.

Żerdzie karmnika zostały oklejone samoprzylepną taśmą miedzianą o szerokości około 6 mm. Przez taśmę i grzędę wywiercony został mały otwór, a do taśmy miedzianej przylutowany został kabelek, który poprowadzono przez otwór i pod karmnikiem. Każdy z tych przewodów jest podłączony do 8-pinowego złącza Dupont. Na **rysunku 4** pokazano gotowy karmnik, uzbrojony w przewodzące taśmy do sensorów pojemnościowych.

Serwer MQTT

Jeśli mieliście kiedyś do czynienia z systemami Internetu Rzeczy (IoT), to być może mieliście do czynienia z MQTT lub być może macie już uruchomiony serwer MQTT w swojej sieci. Jeśli nie, zaleca się użycie Raspberry Pi 3 dla serwera MQTT. Instrukcję uruchomienia i plik obrazu z systemem takiego serwera znaleźć można na stronie Andreasa Spiessa (***http://bit.ly/2Saivob***). Postępowanie zgodnie z informacjami tam zawartymi powoli na przygotowanie serwera MQTT do naszych zastosowań.

Po uruchomieniu serwera Node-Red można zaimportować przepływ monitora karmnika dla ptaków, kopiując dane do *~/RPi_bird_feeder_monitor/json/Bird_Feeder_Monitor_Flow.json* i używając opcji importu ze schowka, aby wkleić dane w nowy przepływ MQTT (**rysunek 5**). Ten przepływ będzie wymagał następujących węzłów:

- **node-red-node-darksky** – API dla DarkSky. Konieczne jest posiadanie konta na portalu DarkSky, aby z niego korzystać. API to zapewnia dane pogodowe dla lokalizacji karmnika. Dostarczane są przez DarkSky i zawierają informacje takie jak m.in. opady, temperaturę, wilgotność, prędkość wiatru oraz dane o zachmurzeniu. Informacja o opadach jest szczególnie ważna, ponieważ służy do ustalenia, czy czujniki pojemnościowe wymagają ponownej kalibracji;
- **node-red-contrib-bigtimer** – Big Timer (timer do synchronizacji z zegarem), opracowany przez Scargill Tech. To szwajcarski scyzoryk wśród timerów – posiada bardzo rozbudowane możliwości. W opisywanym systemie służy do rozpoczynania i zatrzymywania rejestrowania danych o świcie i zmierzchu każdego dnia;
- **node-red-contrib-influxdb** – narzędzia do zapisu danych do bazy danych InfluxDB.

InfluxDB to lekka i łatwa w użyciu baza danych szeregów czasowych. Baza danych automatycznie dodaje znacznik czasu za każdym razem, gdy wstawiamy do niej nowe dane. W przeciwieństwie do baz takich jak SQLite pola nie muszą być z góry zdefiniowane. Są one dodawane automatycznie po wstawieniu danych do bazy.

Fotografia 4. Karmnik uzbrojony w przewodzące taśmy do sensorów pojemnościowych

Konfiguracja Node-Red

Wspomniany powyżej plik JSON załaduje przepływ, który wymaga tylko kilku poprawek w celu dopasowania go do naszych wymagań.

1. Musimy połączyć „MQTT Publish" oraz „monitor/feeder/#" z serwerem MQTT w naszej sieci.
2. Następnie trzeba ustawić szerokość i długość geograficzną naszej lokalizacji w węźle Big Timer „Dawn & Dusk Timer (config)".
3. Skonfiguruj węzeł „monitor/feeder/astronomy (config)". Kamerę można włączyć/wyłączyć dla każdej żerdzi niezależnie. Autor konstrukcji nie włącza ich dla dwóch tylnych, których po prostu kamera nie widzi.
4. Ustaw węzeł „Counter Timer (config)" na żądany przedział czasu. Domyślnie jest to 5 minut.
5. Szerokość i długość geograficzną lokalizacji karmnika podać trzeba również w węźle „DarkSky (config)". Trzeba tutaj także wprowadzić klucz API DarkSky (w darksky-credentials).
6. W węźle funkcyjnym „monitor/feeder/recalibrate (config)" ustawić należy graniczną ilość opadów, która wywoływać będzie rekalibrację czujnika. Domyślnie = 0,001 cala/h.
7. Edytuj węzeł „Topic Filter for MQTT Receiver Debug Node (config)", by usunąć wszystkie wiadomości MQTT, których nie chcemy widzieć.
8. Opcjonalnie: jeśli chcesz przechowywać dane w arkuszu kalkulacyjnym na Dysku Google, musisz edytować węzeł funkcyjny „Build Google Docs Payload (config)" z identyfikatorami pól tego arkusza.
9. Opcjonalnie: Dodaj swój unikalny formularz URL do pola URL węzła „Google Docs GET (config)".

Pulpit interfejsu użytkownika Node-red

Bird_Feeder_Monitor_Flow zawiera interfejs użytkownika, zapewniający dostęp do serwera MQTT za pośrednictwem np. telefonu komórkowego. Monitor można ręcznie włączać lub wyłączać, ponownie skalibrować czujniki lub manualnie zrobić zdjęcia. Pokazana jest również suma „dotknięć" czujnika, co daje przybliżone wyobrażenie o liczbie ptaków odwiedzających karmnik.

Grafana

Jest to pakiet do analizy metrycznej i wizualizacji o otwartym kodzie źródłowym. Jest najczęściej używany do wizualizacji danych dostarczanych w postaci szeregów czasowych. Używa się go w celu monitorowania infrastruktury i aplikacji, ale wiele z aplikacji tego pakietu znajdziemy również w innych dziedzinach, w tym do wyświetlania pomiarów z czujników przemysłowych, stanu automatyki domowej, pogody i procesów technologicznych.

Oprogramowanie to znajduje się w obrazie systemu Andreasa Spiessa, o którym wspomniano powyżej. Użyto go do stworzenia serwera MQTT. Po skonfigurowaniu bazy danych InfluxDB na serwerze MQTT, Grafana może być skonfigurowana do korzystania z tej bazy, by prezentować dane zbierane przez system.

Panel wykresów Grafany, używany w tym projekcie, można załadować z pliku JSON, który znajduje się pod ścieżką *~/Rpi_bird_feeder_monitor/json/Bird_Feeder_Monitor_Grafana.json*.

InfluxDB

W poradniku Andreasa Spiessa opisana jest także baza danych InfluxDB. Znajdziemy tam dokładny opis jak skonfigurować tego rodzaju bazę na naszym komputerze. Poniżej znajduje się krótki opis kroków, jakie trzeba przejść, by taką bazę uruchomić. W pierwszej kolejności logujemy się do systemu poprzez SSH i tworzymy nowego użytkownika Influx – w konsoli Influx (wchodzimy do niej po wpisaniu w linii komend polecenia influx) wpisujemy:

```
CREATE USER „pi” WITH PASSWORD
‘raspberry’ WITH ALL PRIVILEGES
SHOW USERS
```

Co powinno stworzyć użytkownika pi z hasłem raspberry z kompletem uprawnień. Następnie, nie wychodząc z konsoli Influxa, tworzymy pustą bazę danych *BIRD_FEEDER_MONITOR*. W tym celu korzystamy z komendy:

```
CREATE DATABASE BIRD_FEEDER_MONITOR
SHOW DATABASES
```

Po utworzeniu nowej bazy, można skonfigurować InfluxDB w Node-Red. Pomiar można nazwać „karmnik” i do niego zapisywać dane. Po połączeniu Influxa z serwerem MQTT, w bazie danych powinna być widoczna wybrana nazwa pomiarów.

Na tym kończy się konfiguracja bazy danych – dzięki temu, że nie musimy definiować jej pól, nie musimy wprowadzać do niej informacji o tym jakie dane będą podawane. Baza danych automatycznie dopasuje się do tego, co zostanie do niej przesłane.

Kamera z Raspberry Pi

Aby zbudować tego rodzaju kamerę, po zainstalowaniu Rasbiana na komputerze i jego zaktualizowaniu, należy zainstalować git-a, pip-a i paho-mqtt. Następnie możemy sklonować oprogramowanie kamery z repozytorium:

```
cd ~
git clone "https://github.com/sbkirby/RPi_bird_
feeder_monitor.git"
```

Jeśli oprócz zdjęć chcemy także nagrywać filmy, zainstalować musimy także ffmpeg:

```
git clone "https://git.ffmpeg.org/ffmpeg.git" ffmpeg
cd ffmpeg
./configure
make
sudo make install
```

Następnie konfigurujemy uprawnienia w oprogramowaniu Bird Feeder Monitoring:

```
cd RPi_bird_feeder_monitor
sudo chmod 764 make_movie.sh
sudo chmod 764 take_photo.sh
sudo chown www-data:www-data make_movie.sh
sudo chown www-data:www-data take_photo.sh
```

Uruchamianie przy starcie

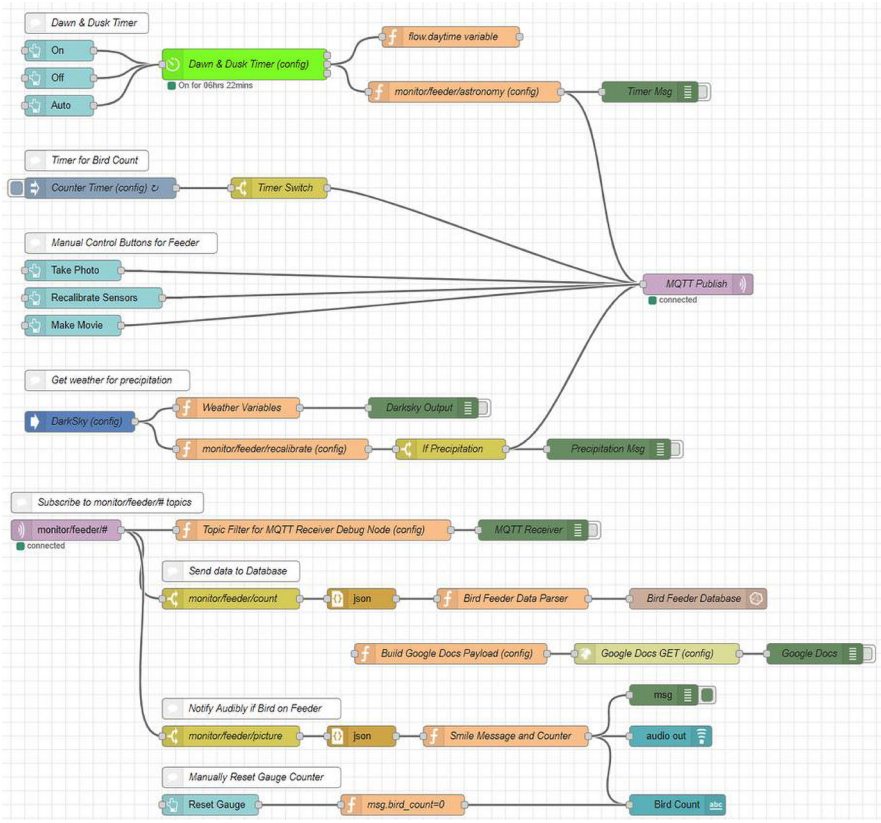
Aby program obsługujący kamerę uruchamiał się automatycznie, musimy stworzyć odpowiedni skrypt uruchamiający skrypt i dodać go do *crona* – narzędzia do automatycznego uruchamiania oprogramowania przez system operacyjny Linux. W pierwszej kolejności tworzymy skrypt:

```
cd RPi_bird_feeder_monitor
nano launcher.sh
```

W skrypcie umieszczamy następującą zawartość:

```
#!/bin/sh
# launcher.sh
# navigate to home directory, then to this directory,
then execute python script, then back home
cd /
cd home/pi/RPi_bird_feeder_monitor
sudo python3 camera_mqtt_client.py
cd /
```

Wyjdź i zapisz *launcher.sh*. Aby plik był wykonywalny nadaj mu odpowiednie uprawnienia:



Rysunek 5. Schemat przepływu MQTT

```
chmod 755 launcher.sh
```

Finalnie tworzymy katalog dla logów:

```
cd ~
mkdir logs
```

Następnie musimy edytować crontaba, aby uruchomić stworzony skrypt podczas uruchamiania. W tym celu w terminalu wpisujemy:

```
sudo crontab -e
```

Spowoduje to wyświetlenie okna *crontab*. Na końcu pliku i dodajemy następujący wiersz.

```
@reboot sh /home/pi/RPi_bird_feeder_monitor/
launcher.sh >/home/ pi/ logs/cronlog 2>&1
```

Wyjdź i zapisz plik, a następnie uruchom ponownie komputer. Skrypt powinien automatycznie uruchomić skrypt *camera_mqtt_client.py* przy starcie systemu. Status skryptu można sprawdzić w plikach dziennika znajdujących się w folderze */logs*.

Podsumowanie

Zaprezentowany system pozwala na podglądanie ptaków, które dokarmiamy w niemalże dowolny sposób. Dzięki zastosowaniu MQTT, system jest gotowy na dalszą rozbudowę o inne moduły. Realizacja projektu z pewnością nauczy nas wiele, w szczególności o korzystaniu InfluxDB i Grafany – dwóch bardzo wartościowych narzędzi programowych, wykorzystywanych w systemach Internetu Rzeczy.

Jak wskazuje sam autor, nie jest to w 100% gotowa konstrukcja, która nie wymaga dalszego dopracowania. O ile samo oprogramowanie jest kompletne i w pełni funkcjonalne, to poprawek wymagać mogą algorytmy używane do analizy danych. „Jedną z rzeczy, które odkryliśmy za pomocą monitora, jest częstotliwość sytuacji, gdy ptak ląduje na jednej grzędzie, a następnie przeskakuje na kolejne, dopóki nie obskoczy całego karmnika”. Z uwagi na sposób wykrywania ptaków i brak dyskryminacji w systemie, powoduje to sztuczne zawyżanie ich ilości.

Nikodem Czechowski

Źródło
http://bit.ly/2Z6uCUN