

POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

KIERUNEK: TELEINFORMATYKA (TIN)

SPECJALNOŚĆ: UTRZYMANIE SIECI TELEINFORMATYCZNYCH (TIU)

PRACA DYPLOMOWA MAGISTERSKA

Wielosystemowa platforma dydaktyczna
dla sieci sensorowej IoT

An educational platform for the IoT multi-
system sensor network

Autor : Tomasz Borusiewicz

Opiekun pracy:
Dr hab. inż. Kamil Staniec

OCENA PRACY:

SPIS TREŚCI

INDEKS SYMBOLI	2
INDEKS SKRÓTOWCÓW	3
1. Wstęp	4
2. Cel i zakres pracy	5
3. Analiza dostępnych systemów służących do realizacji zadań sensorowo-transmisyjnych	5
3.1 LoRa	5
3.2 HC-12	6
3.3 NRF24L01	7
3.4 Xbee	8
3.5 Z-Wave	11
3.6 WiFi	12
3.7 Sigfox	13
3.8 Bluetooth	13
4. Zestawienie platformy sprzętowej	14
4.1 Dobór urządzeń do bezprzewodowego przesyłania danych	14
4.2 Dobór mikrokontrolerów	19
4.3 Połączenie elementów platformy edukacyjnej	23
4.4 Skrypty obsługujące moduły sensorowo-komunikacyjne	29
4.4.1 HC-12	30
5. System archiwizacji i wizualizacji danych	30
6. Maszyna stanów wielosystemowej platformy czujnikowo komunikacyjnej	31
7. Koncepcja badania odporności systemów na zakłócenia???	Error! Bookmark not defined.
8. Koncepcja ćwiczeń laboratoryjnych	34
9. Bibliografia	37
Dodatek A. Kody źródłowe urządzeń Arduino uno, Arduino mega, Adafruit Feather M0, Sparkfun fio v3 oraz Raspberry pi zero w	40

INDEKS SYMBOLI

<i>mA</i>	-	<i>mili Amper</i>
<i>μA</i>	-	<i>mikro Amper</i>
<i>V</i>	-	<i>Volt</i>
<i>mW</i>	-	<i>mili Wat</i>
<i>dBm</i>	-	<i>decybel w odniesieniu do mili wata</i>
<i>mm</i>	-	<i>mili metr</i>
<i>kb/s</i>	-	<i>kilo bit na sekundę</i>
<i>b/s</i>	-	<i>bit na sekundę</i>

INDEKS SKRÓTOWCÓW

IoT	-	Internet rzeczy (ang. Internet of Things)
ID	-	Identyfikator (ang. Identification)
IoMT	-	Internet przedmiotów medycznych (ang. Internet of Medical Things)
UART	-	Uniwersalny asynchroniczny nadajnik-odbiornik (ang. Universal Asynchronous Receiver-Transmitter)
SPI	-	Serialowy interfejs urządzeń peryferyjnych (ang. Serial Peripheral Interface)
USB	-	Uniwersalna magistrala szeregowo (ang. Universal Serial Bus)
GFSK	-	Modulacja z kluczkowaniem częstotliwości w której stosuje się fale elektromagnetyczne o kształcie krzywej Gaussa (ang. Gaussian FSK)
ADC	-	Konwerter analogowo cyfrowy (ang. Analog to Digital Converter)
I2C	-	Szeregowa, dwukierunkowa magistrala służąca do przesyłania danych (ang. Inter-Integrated Circuit)
PWM	-	Modulacja szerokości impulsów (ang. Pulse-Width Modulation)
RAM	-	Pamięć o dostępie swobodnym (ang. Random Access Memory)
OTG	-	Specyfikacja pozwalająca na łączenie urządzeń USB (ang. On The Go)
CSI	-	Szeregowy interfejs kamery (ang. Camera Serial Interface)
LPWAN	-	Rozległa sieć małej mocy (ang. Low Power Wide Area Network)

1. Wstęp

Internet rzeczy (ang. Internet of Things) to system powiązanych ze sobą urządzeń komputerowych, wyposażonych w unikalne identyfikatory (IDs) oraz możliwość wymiany informacji między sobą poprzez sieć teleinformatyczną bez konieczności interakcji człowieka z człowiekiem lub człowieka z maszyną [1]. Pierwsze wzmianki o inteligentnej sieci IoT pojawiają się już w 1982 r., w którym to automat Cola-Coli na Uniwersytecie Carnegie Mellon został podłączony do Internetu [2]. Urządzenie potrafiło wysłać informacje o aktualnym inwentarzu i czy nowo załadowane napoje są zimne czy nie. Termin Internetu rzeczy (ang. Internet of Things) pierwszy raz został użyty przez Kevina Ashtona w trakcie prezentacji dla firmy Procter & Gamble [3] w 1999 r. Z roku na rok liczba urządzeń podłączonych do sieci Internet stale rośnie. Firma Cisco w swoim artykule „How the Next Evolution of the Internet Is Changing Everything” [4] pokazuje, że już w 2003r. liczba ta wynosiła około 500 milionów, a w roku 2010 była już większa (12,5 biliona) niż populacja ludzi na ziemi (6,8 biliona). Przedsiębiorstwo zakłada również, że w roku 2020r. liczba połączonych ze sobą urządzeń wynosić będzie około 50 milionów, co daje 6.58 urządzenia na osobę. IoT znalazło zastosowanie w wielu dziedzinach życia takich jak: automatyka domowa, przemysł motoryzacyjny, ochrona zdrowia, wojsko, transport i wiele innych [5]. Dzięki inteligentnym urządzeniom zainstalowanym w domu użytkownik może zaoszczędzić pieniądze np. poprzez automatyczne wyłączanie i włączanie światła. Zaautomatyzowany dom może być oparty na specjalnych hubach które sterują systemami IoT. Przykładem takiego rozwiązania jest HomeKit firmy Apple, dzięki któremu użytkownik może kontrolować wszystkie urządzenia poprzez telefon z systemem IOS [6]. Oprócz systemów komercyjnych istnieją również rozwiązania typu open source, takie jak Home Assistant, OpenHAB czy Domoticz [7], które w prosty sposób pomagają zarządzać urządzeniami składającymi się na inteligentny dom. Jednym z kluczowych zastosowań automatyzacji domu jest zapewnienie pomocy osobom niepełnosprawnym i starszym. Systemy te wykorzystują technologię wspomagającą w celu dostosowania się do niepełnosprawności właściciela [8]. Sterowanie głosowe może pomóc użytkownikom w ograniczeniu wzroku i mobilności, podczas gdy systemy alarmowe mogą być podłączone bezpośrednio do implantów ślimakowych noszonych przez użytkowników z upośledzeniem słuchu [9]. Kolejną bardzo przydatną dziedziną życia w której rozwinął się przemysł IoT jest medycyna. Internet przedmiotów medycznych (IoMT) opisuje urządzenia związane ze zdrowiem, gromadzeniem i analizowaniem danych do badań [10]. IoMT jest określane również jako „Smart Healthcare” [11].

2. Cel i zakres pracy

3. Analiza dostępnych systemów służących do realizacji zadań sensorowo-transmisyjnych

Intensywny rozwój Internetu Rzeczy (IoT) powoduje, że coraz częściej urządzenia wykorzystują bezprzewodową łączność, zaczynając na elektronice osobistej, poprzez rozmaite systemy czujnikowe aż do zaawansowanych układów przemysłowych. Zróżnicowanie urządzeń, sposób konstrukcji, warunku pracy oraz ich przeznaczenie wymusza na producentach stosowanie różnych systemów komunikacji, najlepiej dopasowanych do ściśle określonych potrzeb klienta. Wybierając rozwiązania bezprzewodowej łączności warto zwrócić uwagę na kilka aspektów, nie zawsze najlepszym systemem będzie ten o największym zasięgu czy przepustowości danych. Ważne są również koszt danego systemu, jego wytrzymałość w określonych warunkach pracy czy czas działania bez konieczności wymiany zasilania. Todo: jeszcze parę zdań napisać

3.1 LoRa

LoRa (ang. Long Range) to radiowy protokół komunikacji dalekiego zasięgu umożliwiający urządzeniom dostęp do Internetu, przy utrzymaniu niskiego poboru energii (LPWAN). Kładzie nacisk na komunikację dalekiego zasięgu z wysoką czułością odbioru sygnału, dzięki temu pozwala na efektywną pracę w silnie zakłócanym środowisku. Technologia ta daje możliwość uzyskania do 5-letniego okresu pracy na baterii. LoRa korzysta z darmowych pasm częstotliwości ISM 433 MHz (Asia), 863-870 MHz (Europa), 902-928 MHz (Ameryka Północna), 915-928 (Australia). Protokół bazuje na zmodyfikowanej modulacji CSS (ang. Chirp Spread Spectrum), która korzysta z techniki rozpraszania widma[11]. Technika modulacji zastosowana w LoRa sprawia, że jest odporna na szumy kanałowe, ponieważ cała przydzielana szerokość pasma jest wykorzystywana do nadania sygnału. Dodatkowo transmisja rozproszona jest w sposób pseudolosowy, który odbierany jest jako szum, co gwarantuje bezpieczeństwo przesyłanych danych[12]. Technologia opracowana została przez firmę Semtech i aktualnie rozwijana jest przez grupę LoRa Alliance, która zrzesza ponad 500 członków m.in. wspomniana wcześniej firma Semtech ale również Cisco czy Amazon. Dzięki wyżej wymienionym korzyści producenci urządzeń coraz częściej korzystają z rozwiązań LoRa.



Rys. 3.1 - Mikrokontroler Feather M0 z modułem RFM9x LoRa firmy Adafruit

Moduł Feather M0 firmy Adafruit. Posiada wbudowane 32 - bitowy mikrokontroler ATSAM21G18 ARM Cortex M0, dzięki czemu nie ma potrzeby podłączać go do zewnętrznego urządzenia sterującego. Dedykowane złącze dla baterii Lipol pozwala na łatwe zasilanie urządzenia z zewnętrznego źródła.

3.2 HC-12

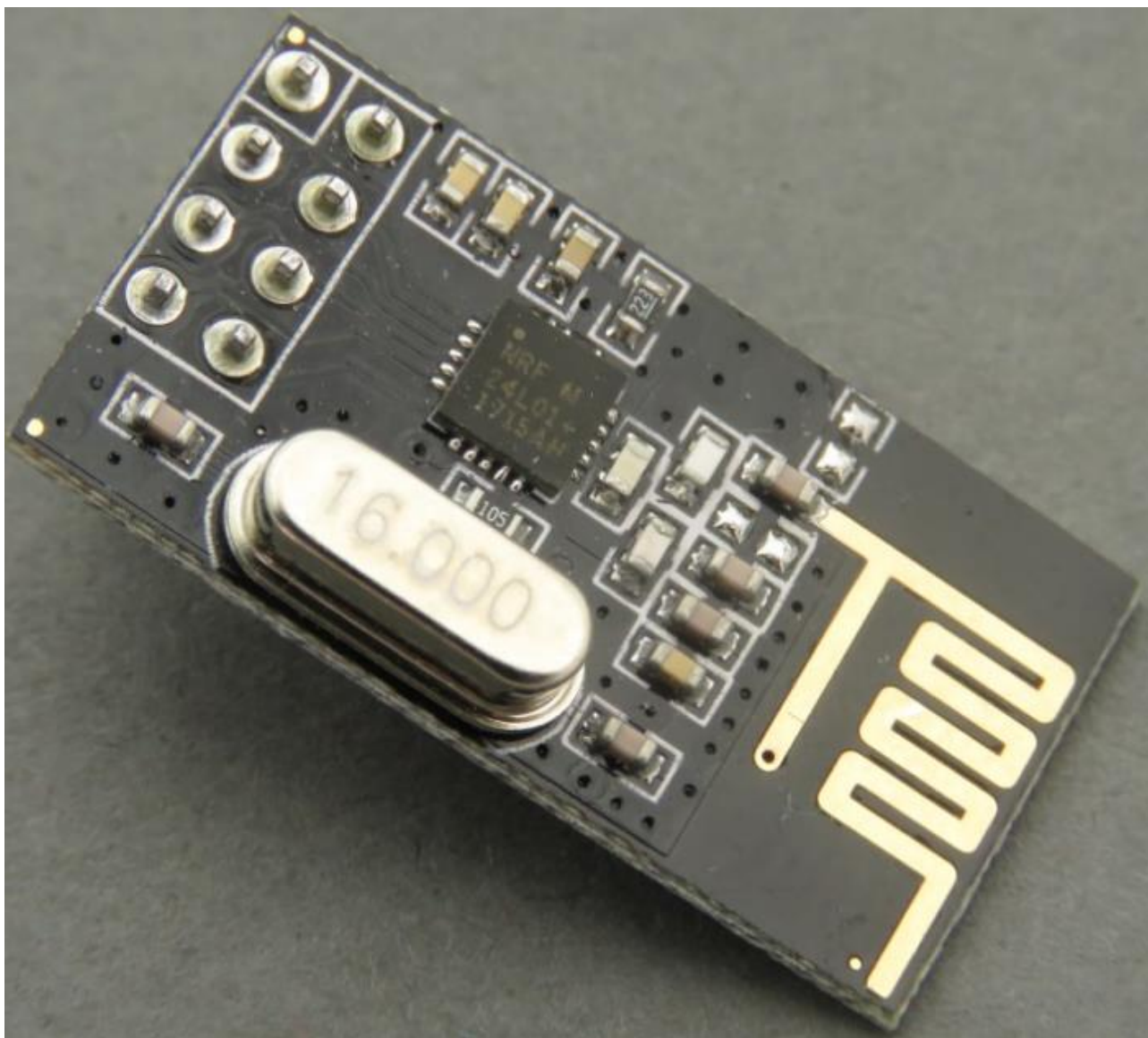
Moduł łączności bezprzewodowej działający na częstotliwości 433,4 MHz do 473 MHz z możliwością konfiguracji do 100 oddzielnych kanałów o szerokości 400 KHz. Maksymalna moc nadawania sygnału wynosi 100 mW (20 dB), czułość odbiornika to -117 dBm przy transmisji z przepustowością 5000 b/s. Zasięg komunikacji to 1000 metrów w otwartej przestrzeni. Minimalny pobór prądu wynosi 80 μ A, maksymalny 100 mA.



Rys. 3.2 - Moduł HC-12

3.3 NRF24L01

NRF24L01 jest jedno czipowym nadajnik-o odbiornikiem radiowym działającym w dsfsdsgólnościowym paśmie ISM 2,4 GHz. Umożliwia pracę trybie niskiego poboru mocy do 26 μA w trybie czuwania oraz do 900 nA w trybie power down. Moduł pozwala na konfigurację 126 kanałów radiowych w modulacji GFSK z prędkościami 250 kb/s, 1 Mb/s oraz 2 Mb/s. Urządzenie zasilane jest prądem o napięciu od 1,9 V do 3,6 V. Komunikacja z mikrokontrolerem odbywa się poprzez interfejs SPI.



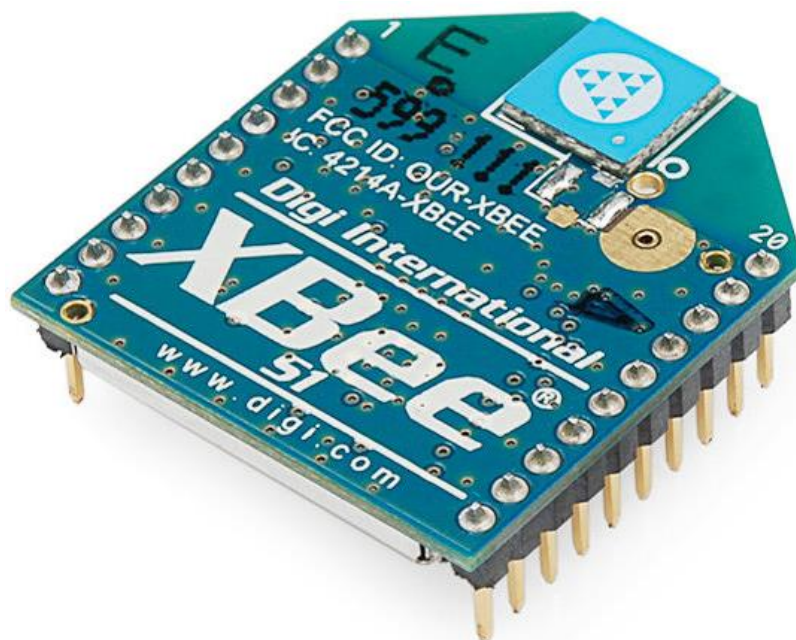
Rys. 3.3 - Moduł NRF24L01

3.4 Xbee

Moduły Xbee to jedno z wielu rozwiązań komunikacji bezprzewodowej opartej na protokole Zigbee oraz standardach opisanych w dokumencie IEEE 802.15.4[17]. Urządzenia pozwalają na tworzenie sieci typu punkt punkt, punkt wielopunkt, mesh oraz inne. Xbee zostały zaprojektowane przede wszystkim do aplikacji wymagających dużego ruchu danych, małych opóźnień i przewidywalnej synchronizacji komunikacji[18], dzięki czemu są dobrym wyborem do pracy w środowiskach mieszkalnych, edukacyjnych czy szpitalnych. Technologia Zigbee opisuje 3 węzły sieci:

- Koordynator Zigbee: Odpowiedzialny za zarządzanie całą siecią, a także przepływ i adresowanie wiadomości oraz ustanowienie identyfikatora sieci (ang. PAN ID) dla całej sieci. W sieci może być tylko jeden koordynator.
- Router Zigbee: Funkcyjny węzeł, który łączy węzły końcowe z węzłami koordynującymi i umożliwia rozbudowę sieci pod względem zakresu i zasięgu.
- Urządzenie końcowe Zigbee (węzeł końcowy): Mogą komunikować się z węzłami koordynującymi bezpośrednio lub za pomocą węzła routera. Jego głównym zastosowaniem jest odbiór danych urządzeń mierzących oraz uruchamianie na ostatnim poziomie aplikacji, w której używana jest sieć

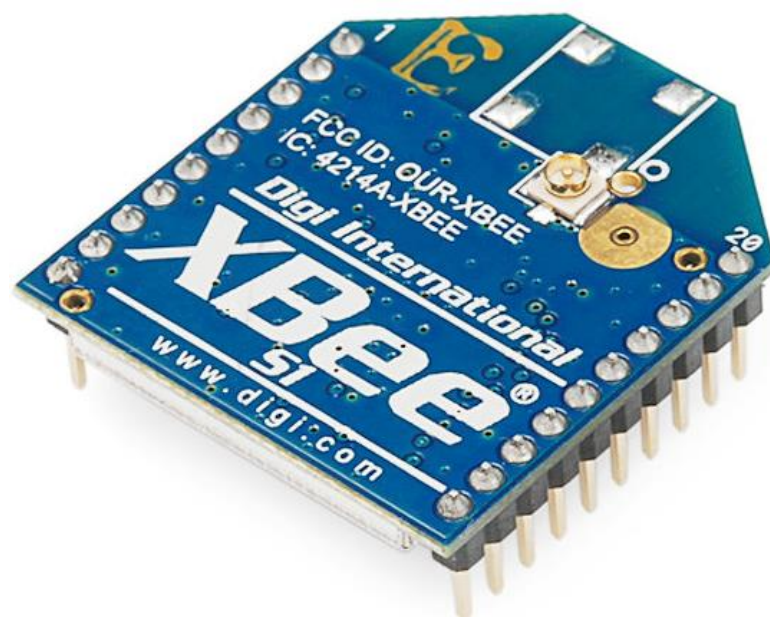
Technologia zapewnia transmisję na pasmach 2,4 GHz (cały świat) i 865 MHz, 915 MHz (wybrane rejony świata). Moduły Xbee dostarczane są różnymi typami anten: chip [Rys. 1.4], wire [Rys. 1.5], u.FL [Rys. 1.6], RPSMA [Rys. 1.7], Trace (PCB) [Rys. 1.8].



Rys. 3.4 - Moduł Xbee z anteną typu chip.



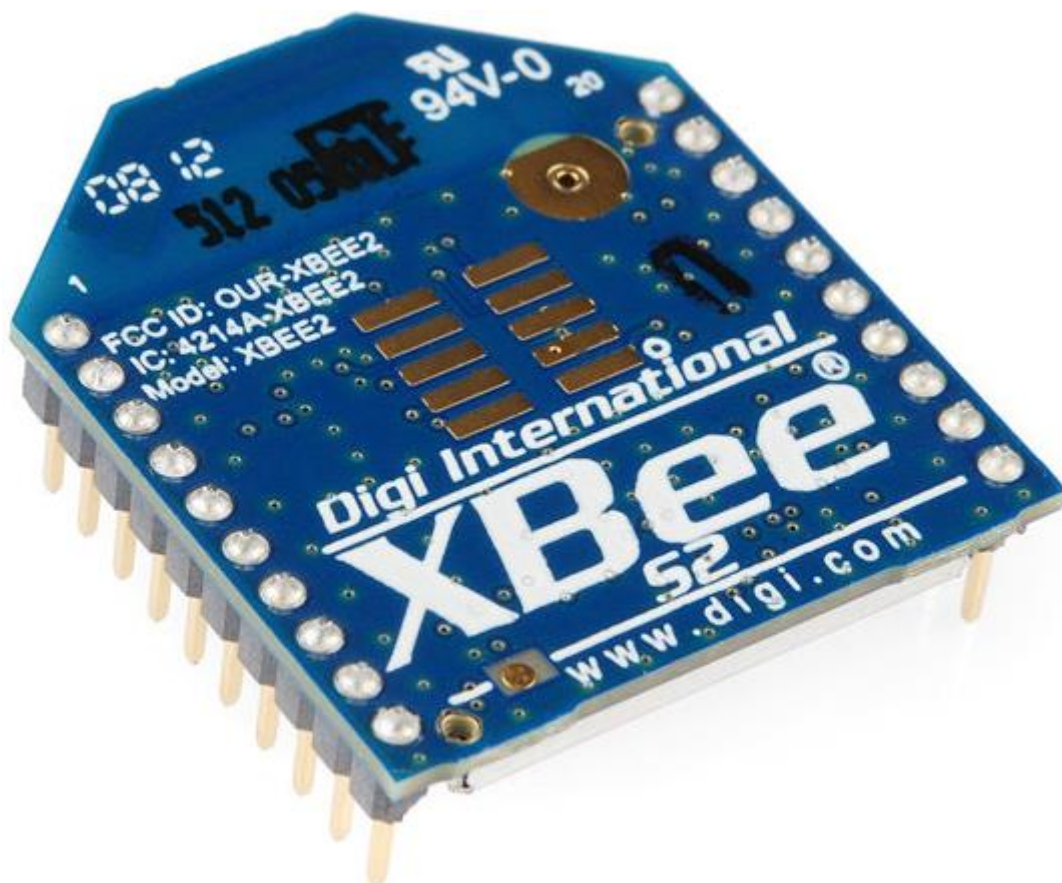
Rys. 3.5 - Moduł Xbee z anteną typu wire.



Rys. 3.6 - Moduł Xbee z anteną typu u.FL.



Rys. 3.7 - Moduł Xbee z anteną typu RPSMA.



Rys. 3.8 - Moduł Xbee z anteną typu Trace (PCB).

3.5 Z-Wave










Z-Wave to protokół stworzony do zdalnego sterowania urządzeniami wykorzystywanymi w systemach inteligentnego zarządzania budynkiem. Ideą systemu jest synchronizacja węzłów sieci w topologię mesh, zwaną również jako sieć kratowa. Każdy węzeł sieci komunikuje się z centralą sieci oraz jest jednocześnie przekaźnikiem danych od innych urządzeń. Umożliwia to budowę sieci o wiele większą niż zasięg połączeniami punkt - punkt. Jedynymi wyjątkami są urządzenia Z-Wave zasilane bateryjnie, które ze względu na potrzebę niskiego poboru prądu mają wyłączoną możliwość przekazywania informacji od innych węzłów. W przeciwieństwie do innych wymienionych systemów komunikacji wymienionych w niniejszym dokumencie Z-Wave jest zastrzeżonym protokołem, który jest obecnie własnością Sigma Designs i promowany jest przez Z-Wave Alliance. Protokół nie jest kompatybilny z protokołem IP, dlatego urządzenia Z-Wave nie mogą bezpośrednio połączyć się z Internetem lub typowymi urządzeniami użytkownika (np. telefon, laptop). W tym celu wykorzystuje się kontrolery, które pozwalają na interakcje z urządzeniami Z-Wave przez telefon, tablet[20].



Rys. 3.9 - Czujnik ruchu, temperatury i natężenia światła z modulem komunikacyjnym Z-Wave.

3.6 WiFi

Bardzo popularna technologia łączności bezprzewodowej opartej o standardy IEEE 802.11.X, powszechnie wykorzystywana do tworzenia lokalnych sieci komputerowych z dostępem do Internetu. W 2017r. pojawił się standard 802.11ah (zwany także Wi-Fi HaLow), który wykorzystuje darmowe pasmo ISM 900 MHz co pozwala osiągać większy zasięg. Korzysta również z niższego zużycia energii, umożliwiając tworzenie dużych grup czujników, które współpracują w celu wymiany informacji, wspierając koncepcję Internetu rzeczy[21].

Wi-Fi HaLow™ for IoT	
Features	Benefits
 Sub-1 GHz spectrum operation	 Long range: approximately 1 km
 Narrow band OFDM channels	 Penetration through walls and other obstacles
 Several device power saving modes	 Supports coin cell battery devices for months or years
 Native IP support	 No need for proprietary hubs or gateways
 Latest Wi-Fi® security	

Rys. 3.10 - Zalety technologii WiFi HaLow [źródło: <https://www.wi-fi.org/discover-wi-fi/wi-fi-halow>]



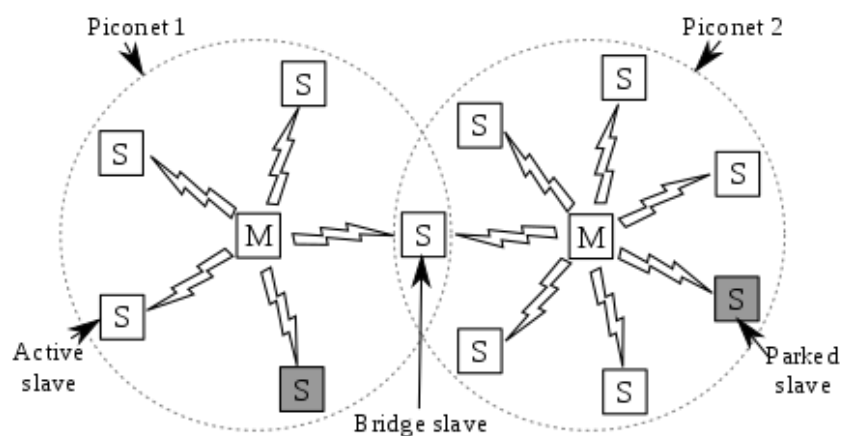
Rys. 3.11 - Moduł bezprzewodowy kompatybilny z 802.11ah HaLow [źródło: <https://www.silextechnology.com/unwired/industry-first-802.11ah-halow-wireless-module-for-iot-devices>]

3.7 Sigfox

Todo: napisać coś albo wywalić ten standard!!!

3.8 Bluetooth

Standard technologii bezprzewodowej stosowanym do wymiany informacji między urządzeniami na krótkim zasięgu. Bluetooth działa na falach centymetrowych w zakresie od 2,402 GHz do 2,480 GHz. Technologia szeroko jest wykorzystywana w urządzeniach takich jak laptopy, smartfony, klawiatury, myszki, bezprzewodowe słuchawki oraz radia. Jednostką standardu jest pikosieć (ang. piconet), w której znajdują się urządzenie typu master z dołączonymi do niego urządzeniami typu slave. W jednym miejscu pracować może wiele pikosieci, które mogą być połączone ze sobą za pomocą węzła typu most (ang. bridge)(Rys 3.12). W sieci mogą znajdować się także węzły w stanie uśpienia (parked slave), które w danym momencie nie wymieniają informacji z masterem ale mogą zostać wybudzone przez urządzenie master.



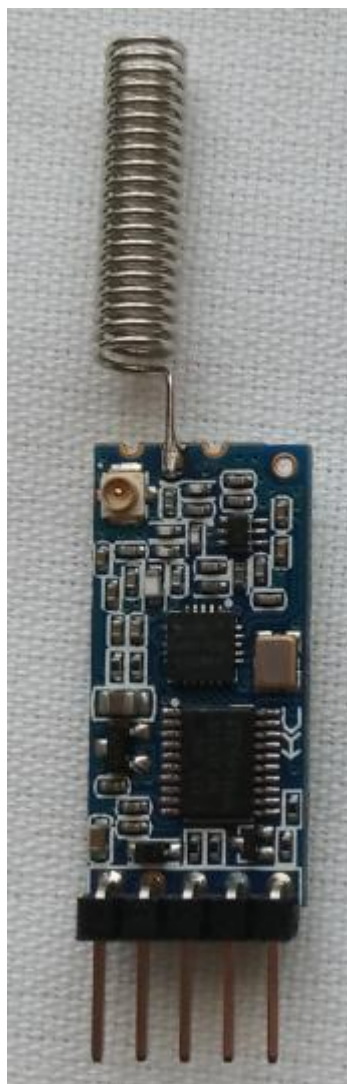
Rys. 3.12 - Architektura sieci Bluetooth

4. Zestawienie platformy sprzętowej

Stworzenie platformy dydaktycznej wymagało dobranie urządzeń, dzięki którym można przesyłać dane w sposób bezprzewodowy, oraz mikrokontrolerów, które pozwalają na interakcje z modułami komunikacji bezprzewodowej. Todo: napisać coś jeszcze

4.1 Dobór urządzeń do bezprzewodowego przesyłania danych

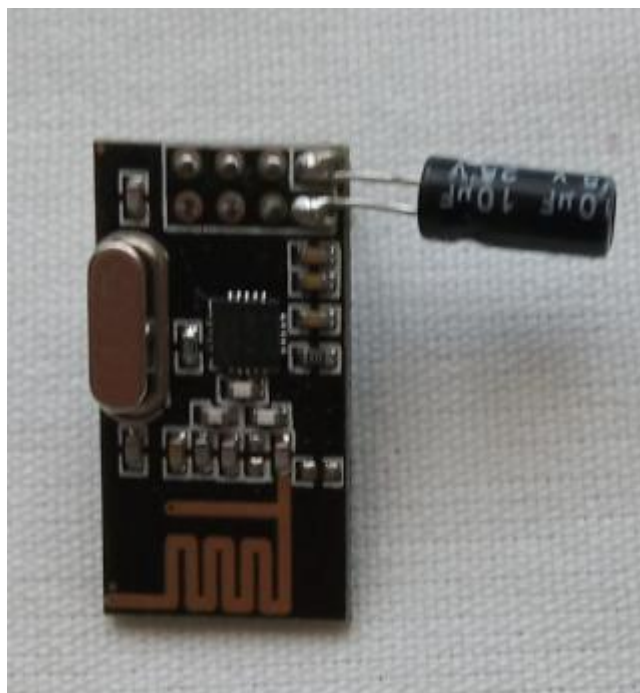
Spośród dużej liczby modułów dostępnych na rynku wybrane zostały cztery, są to: HC-12, NRF24L01, Adafruit Feather M0 z modułem radiowym LoRa, Xbee S1 pro.



Rys. 4.1 - Moduł HC-12

Moduł HC-12 jak już wcześniej wspomniano to urządzenie do realizacji komunikacji bezprzewodowej o zasięgu do 1000 metrów. Szczegółowe informacje[15]:

- Napięcie zasilania: od 3,2 V do 5,5 V
- Pobór prądu:
 - W stanie spoczynku: 20mA
 - Podczas transmisji: 100mA
- Częstotliwość pracy: 433,4 MHz – 473 MHz
- Czulość odbiornika: do -117 dBm
- Moc: do 100 mW (20 dB)
- Komunikacja poprzez interfejs szeregowy UART
- Wymiary: 27,8 x 14,4 x 4 mm



Rys. 4.2 - Moduł NRF24L01

Kolejnym wybranym modułem jest NRF24L01. Urządzenie pracuje w paśmie 2,4 GHz i pozwala na przesyłanie danych z przepustowością do 2 Mb/s. Szczegółowe informacje[16]:

- Napięcie zasilania: od 1,9 V do 3,6 V
- Pobór prądu:
 - W stanie spoczynku: 900µA
 - Podczas transmisji: 11,3mA
- Częstotliwość pracy: 2,400 GHz – 2,485 GHz
- Czulość odbiornika: do -94 dBm
- Moc: do 100 mW (20 dB)
- Komunikacja poprzez interfejs SPI
- Wymiary: 30 x 16 x 3 mm



Rys. 4.3 - Mikrokontroler Adafruit Feather M0 z modułem radiowym LoRa

Następne urządzenie to Adafruit Feather M0 z modułem radiowym LoRa. To urządzenie to integrowany mikrokontroler z modułem LoRa, dzięki czemu może być programowany bezpośrednio poprzez interfejs micro USB bez potrzeby podłączania go do zewnętrznego mikrokontrolera. Szczegółowe informacje[22]:

- Napięcie zasilania: 5 V
- Pobór prądu:
 - W stanie spoczynku: 1,6mA
 - Podczas transmisji: do 120mA
- Częstotliwość pracy: 433,4 MHz – 473 MHz
- Czulość odbiornika: do -140 dBm
- Moc: do 100 mW (20 dB)
- Komunikacja poprzez interfejs UART
- Wymiary: 51 x 23 x 8 mm



Rys. 4.4 - Moduł XBee Pro S1

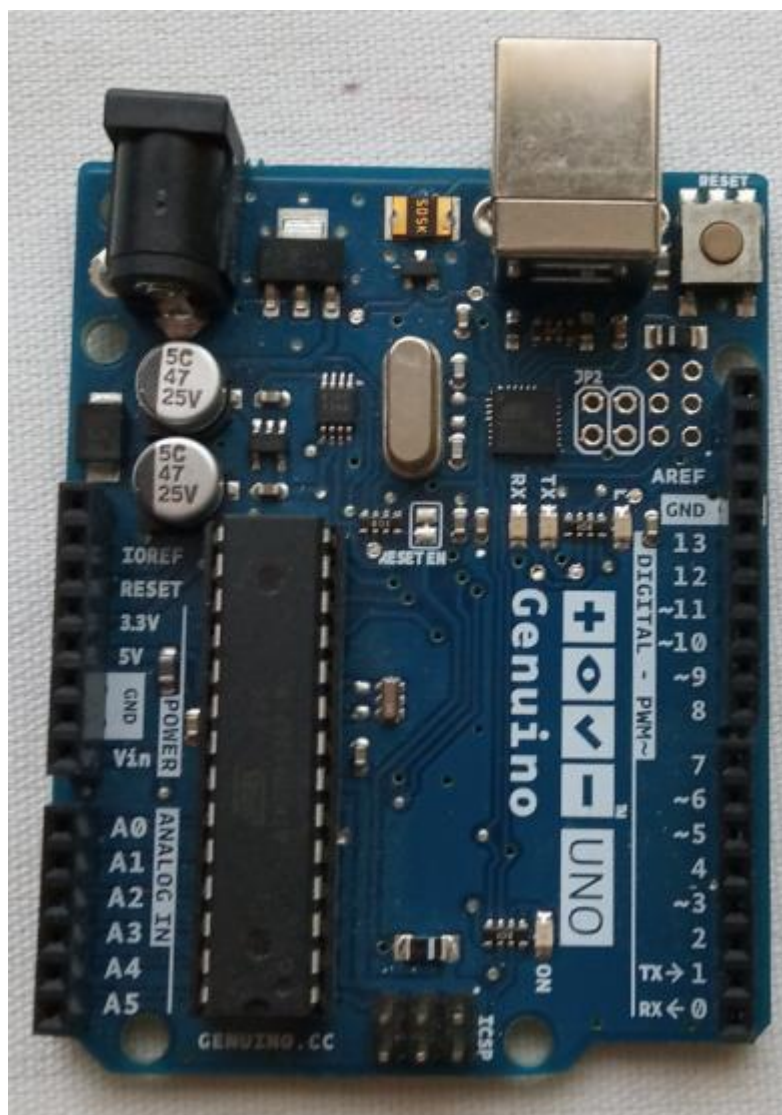
Ostatni wybrany moduł to XBee Pro S1. Moduł bezprzewodowy obsługujący standard IEEE 802.15.4 (ZigBee). Pracuje w paśmie 2.4 GHz z mocą do 60 mW. Szczegółowe informacje[23]:

- Napięcie zasilania: od 2,8 V do 3,4 V
- Pobór prądu:
 - W stanie spoczynku: 10μA
 - Podczas transmisji: do 45mA
- Częstotliwość pracy: 2,400 GHz – 2,485 GHz
- Czułość odbiornika: do -140 dBm
- Moc: do 60 mW (18 dB)
- Komunikacja poprzez interfejs UART
- Wymiary: 33 x 25 x 4,1 mm

Każdy z wymienionych modułów został użyty w dwóch egzemplarzach, jeden jako nadajnik, drugi jako odbiornik. Urządzenia HC-12 oraz NRF24L01 nie mogą być programowane bezpośrednio z poziomu komputera dlatego podłączone zostały do mikrokontrolerów, które pozwalały na interakcje użytkownika z modułami. XBee Pro S1 może być podłączony do komputera ale jest to dość trudne do realizacji i z tego powodu użyto do nich specjalnie dostosowane do nich mikrokontrolery, dzięki którym w wygodny sposób można je konfigurować.

4.2 Dobór mikrokontrolerów

Moduły HC-12 oraz NRF24L01 mogą być konfigurowane za pomocą mikrokontrolerów, również XBee Pro S1 z uwagi na wygodne konfigurowanie podpięte zostały do mikrokontrolera. Urządzenia wybrane do realizacji pracy to Arduino Uno, Arduino Mega 2560, Sparkfun fio v3 oraz Raspberry pi zero w.



Rys. 4.5 - Mikrokontroler Arduino Uno

Arduino Uno to najpopularniejszy mikrokontroler firmy Arduino. Posiada szereg interfejsów komunikacyjnych, takich jak ADC, UART, SPI, I2C. Szczegółowe informacje[24]:

- Napięcie zasilania: od 7V do 12V
- Oparty na mikrokontrolerze ATmega328:
 - Maksymalna częstotliwość zegara 16MHz
 - Pamięć SRAM: 2KB
 - Pamięć FLASH: 32KB
 - Pamięć EEPROM: 2KB
- 16 wejść/wyjść ogólnego przeznaczenia

- 6 wejść analogowych (ADC) o rozdzielczości 10 bitów
- Interfejsy szeregowe: UART, SPI, I2C



Rys. 4.6 - Mikrokontroler Arduino Mega 2560

Arduino Mega 2560 to jedna z najbardziej wydajnych urządzeń firmy Arduino. Posiada 54 cyfrowe interfejsy, 16 interfejsów analogowych. Układ taktowany jest zegarem o częstotliwości 16 MHz. Szczegółowe informacji[25]:

- Napięcie zasilania: od 7V do 12V
- Oparty na mikrokontrolerze ATmega2560:
 - Maksymalna częstotliwość zegara 16MHz
 - Pamięć SRAM: 8KB
 - Pamięć FLASH: 256KB
 - Pamięć EEPROM: 4KB
- 54 wejść/wyjść ogólnego przeznaczenia
- 15 kanałów PWM
- 16 wejść analogowych (ADC) o rozdzielczości 10 bitów
- Interfejsy szeregowe: 4xUART, SPI, I2C



Rys. 4.7 - Mikrokontroler SparkFun Fio v3

Mikrokontroler SparkFun Fio v3 posiada gniazdo na moduł XBee co pozwala na szybkie i proste konfigurowanie go z poziomu komputera. Urządzenie ma również bootloader kompatybilny z Arduino, dzięki czemu można programować go w języku programowania wykorzystywanym w Arduino. Szczegółowe informacje[26]:

- Napięcie zasilania: 5V
- Oparty na mikrokontrolerze ATmega32U4:
 - Maksymalna częstotliwość zegara 16MHz
 - Pamięć SRAM: 2.5KB
 - Pamięć FLASH: 32KB
 - Pamięć EEPROM: 1KB
- 12 wejść/wyjść ogólnego przeznaczenia
- 6 wejść analogowych (ADC) o rozdzielczości 10 bitów
- Interfejsy szeregowo: UART, SPI



Rys. 4.8 - Mikrokontroler Raspberry pi zero w

Raspberry pi to popularny mikrokontroler do wszelakich zastosowań. W niniejszym projekcie, dzięki wbudowanemu modułowi WiFi, posłuży do przesyłania danych otrzymanych z innych mikrokontrolerów do bazy danych. Urządzenie działa na procesorze Broadcom BCM2835 ARM11 pracującym z częstotliwością 1 GHz. Ma wbudowane 512 mega bajtów pamięci operacyjnej RAM oraz szereg interfejsów służących do komunikacji z urządzeniami peryferyjnymi. Szczegółowe informacje[27]:

- Napięcie zasilania: 5V
- Oparty na procesorze Broadcom BCM2835 ARM11:
 - Taktowanie zegara: 1GHz
- 512 MB pamięci RAM
- Moduł WiFi 150 Mb/s 802.11 b/g/n
- Układ Bluetooth Low Energy BLE 4.1
- 40 wejść/wyjść ogólnego przeznaczenia
- Interfejsy szeregowo: UART, SPI, I2C
- Złącze microUSB OTG
- Złącze kamery CSI

Dodatkowo w projekcie wykorzystano czujniki DHT22, które dokonywały pomiaru temperatury otoczenia. Dane zbierane z czujników przesyłane były poprzez systemy komunikacji bezprzewodowej do bazy danych a następnie przekazywane były do systemu wizualizacji danych.



Rys. 4.9 - Czujnik temperatury i wilgotności powietrza DHT22

Czujnik DHT22 to popularny urządzenie do pomiaru temperatury i wilgotności powietrza. Jego napięcie zasilania wynosi od 3,3 V do 6 V co pozwala na podłączenie go do popularnych mikrokontrolerów takich Arduino lub Raspberry pi. Szczegółowe informacje[28]:

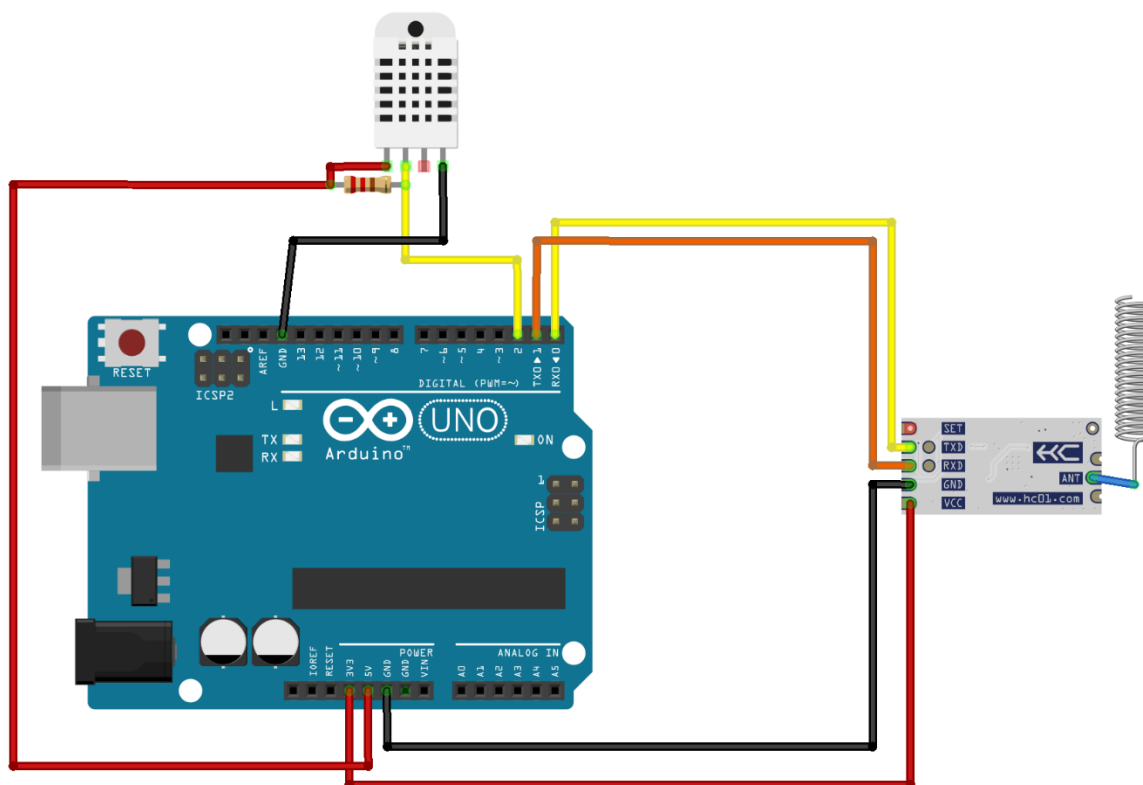
- Napięcie zasilania: od 3,3V do 6V
- Średni pobór prądu: 0,2mA
- Temperatura:
 - Zakres pomiarowy: od -40°C do 80°C
 - Rozdzielczość: 8-bitów (0,1°C)
 - Dokładność: 0,5°C
 - Czas odpowiedzi: 2s
- Wilgotność
 - Zakres pomiarowy: od 0 do 100

4.3 Połączenie elementów platformy edukacyjnej

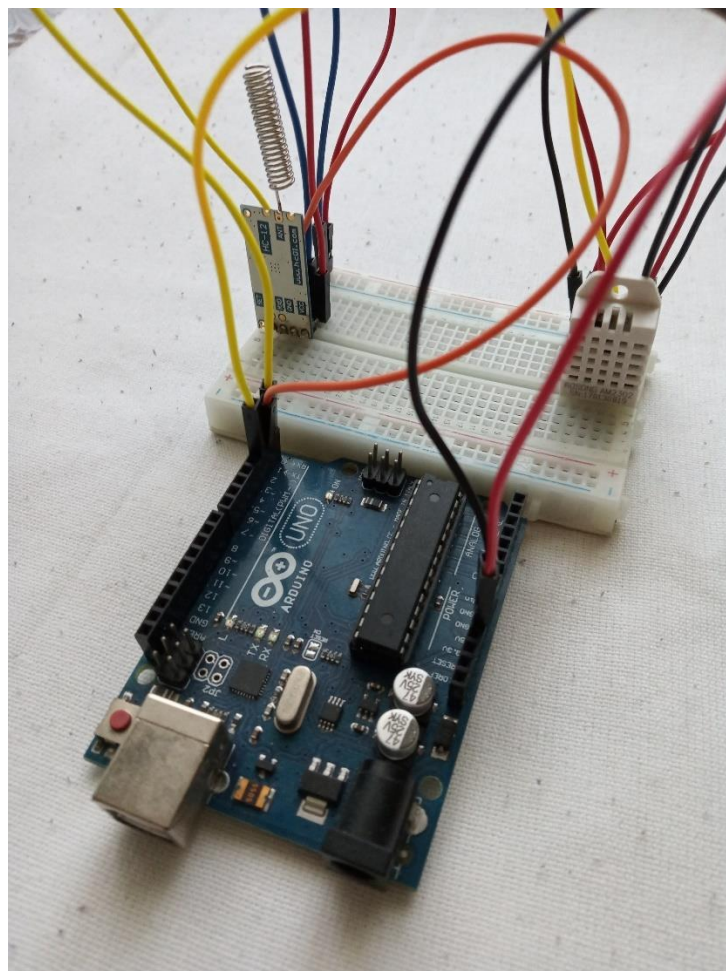
Następnym krokiem zestawiania platformy sprzętowej było fizyczne połączenie komponentów.

Urządzenia połączono ze sobą za pomocą przewodów połączeniowych (ang. Jumper Wire) ze złączami typu męskiego i żeńskiego typu goldpin, które idealnie nadają się do łączenia ze gniazd wlutowanych w wykorzystywanych sprzęcie.

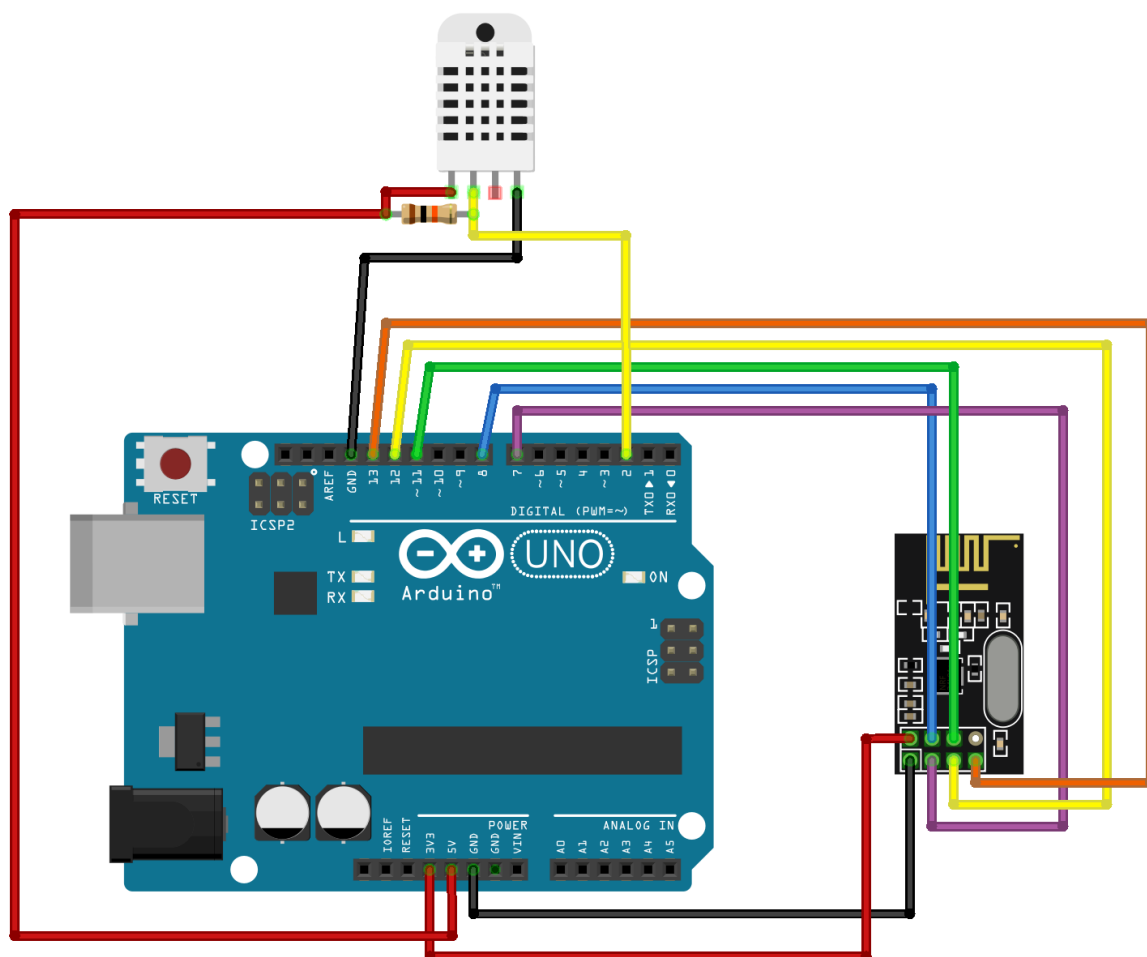
Moduły HC-12 oraz NRF24L01 pełniące rolę nadajnika oraz dwa czujniki DHT22 zostały podłączone do mikrokontrolera Arduino uno jak pokazano na rysunku 4.10, 4.11, 4.12, 4.13. Urządzenia Adafruit Feather M0 z modułem LoRa i mikrokontroler z modułem Xbee zestawione zostały z czujnikami temperatury(Rys. 4.14)(Rys. 4.15)(Rys. 4.16)(Rys. 4.17)



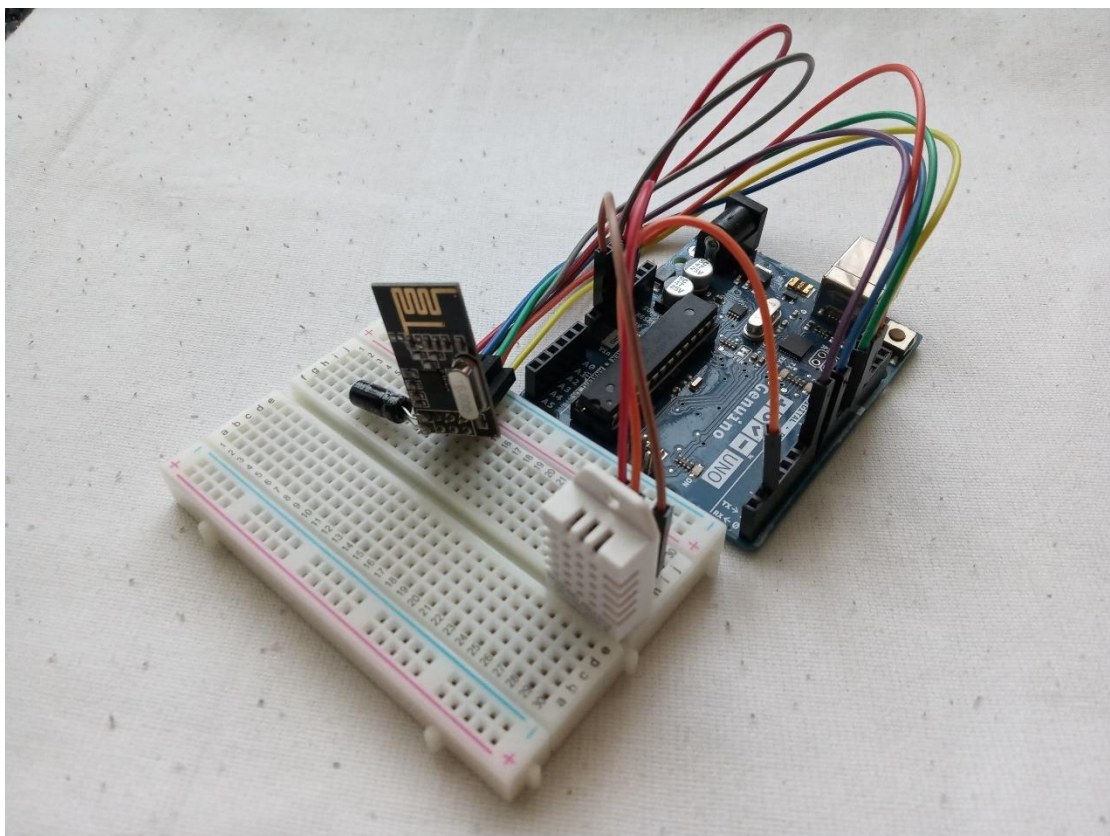
Rys. 4.10 - Schemat połączenia Arduino uno z modułem HC-12 oraz czujnikiem DHT22



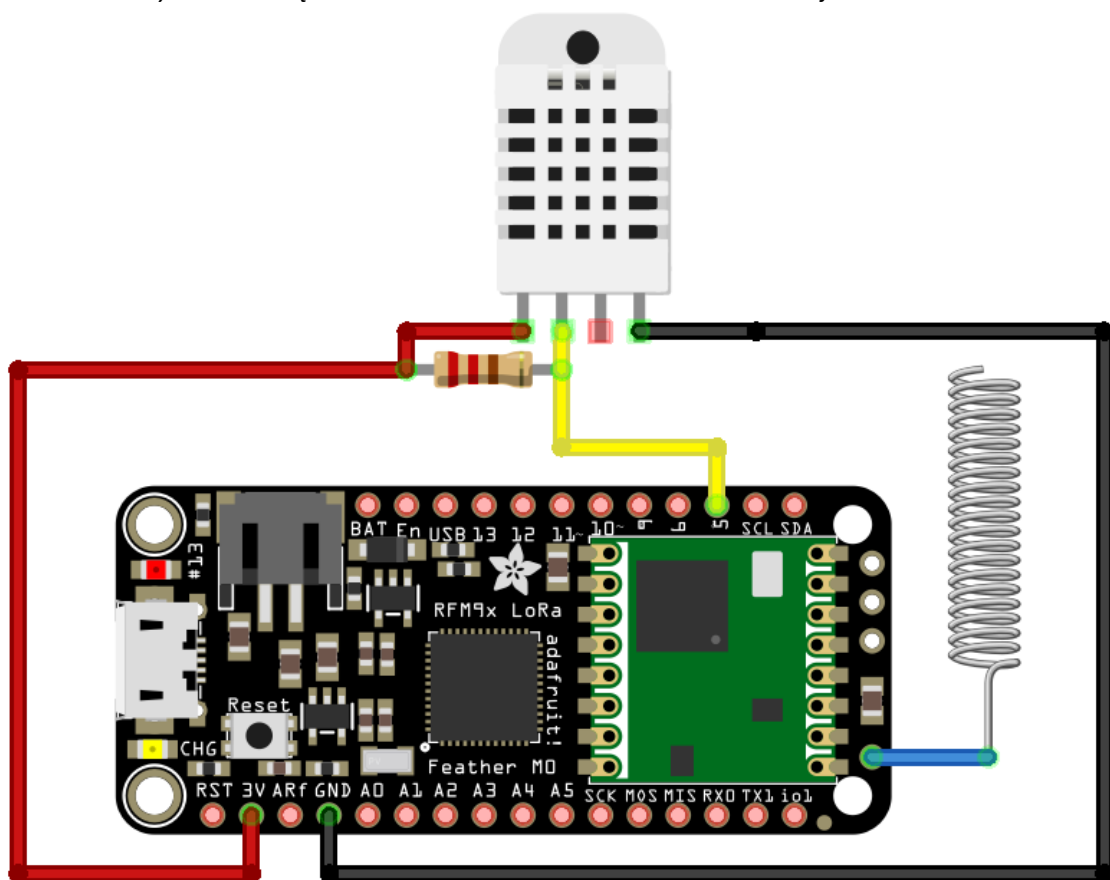
Rys. 4.11 - Połączenie Arduino uno z modułem HC-12 oraz czujnikiem DHT22



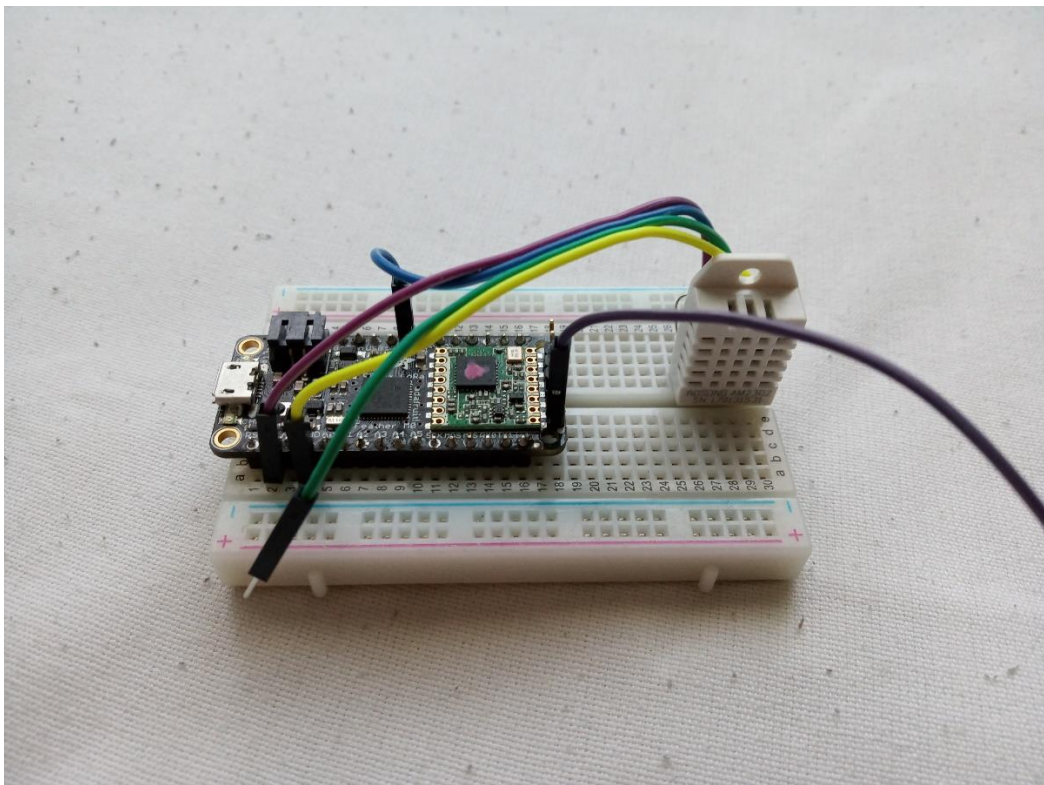
Rys. 4.12 - Schemat połączenie Arduino uno z modułem NRF24L01 oraz czujnikiem DHT22



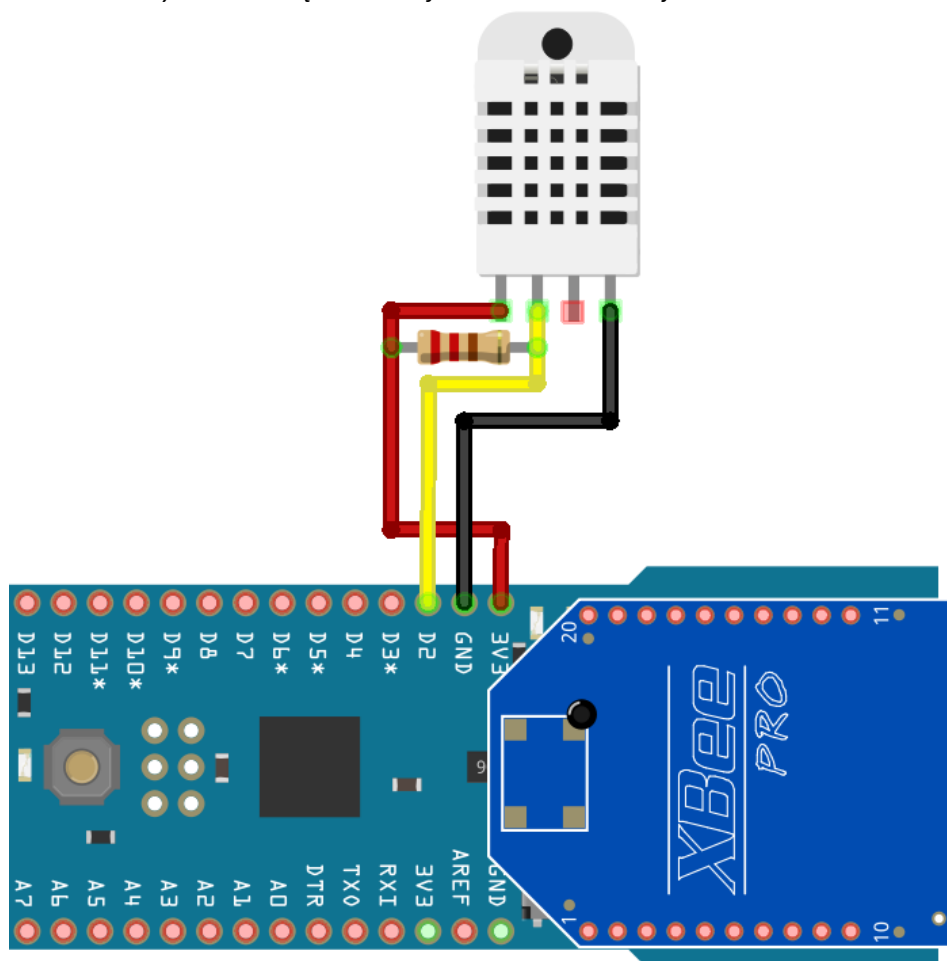
Rys. 4.13 - Połączenie Arduino uno z modułem NRF-L01 oraz czujnikiem DHT22



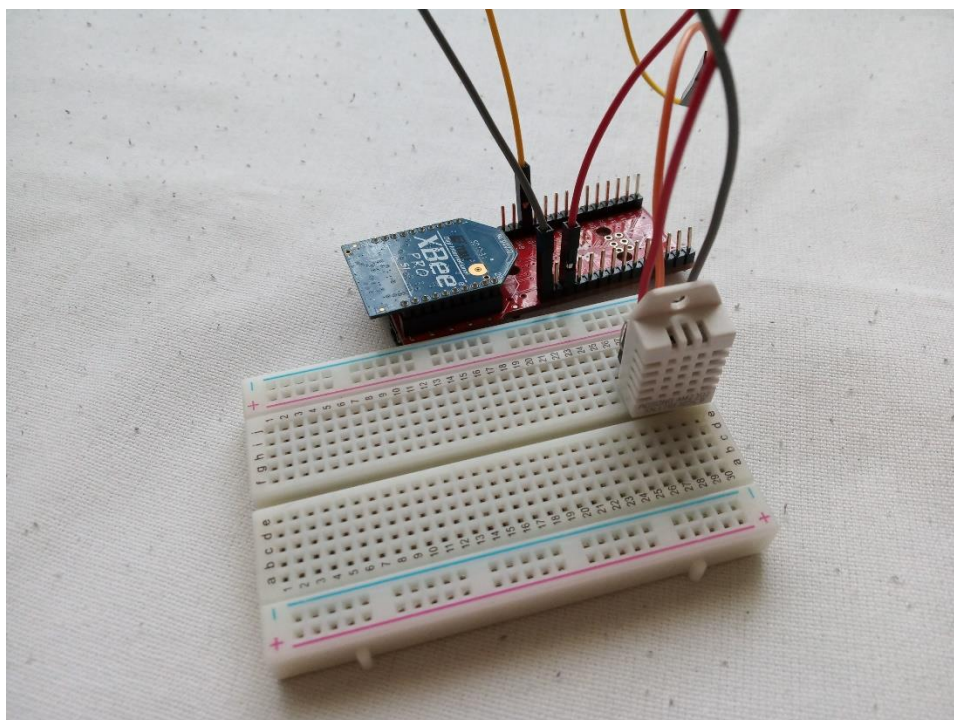
Rys. 4.14 - Schemat połączenia Adafruit Feather M0 z czujnikiem DHT22



Rys. 4.15 - Połączenie Adafruit Feather M0 z czujnikiem DHT22

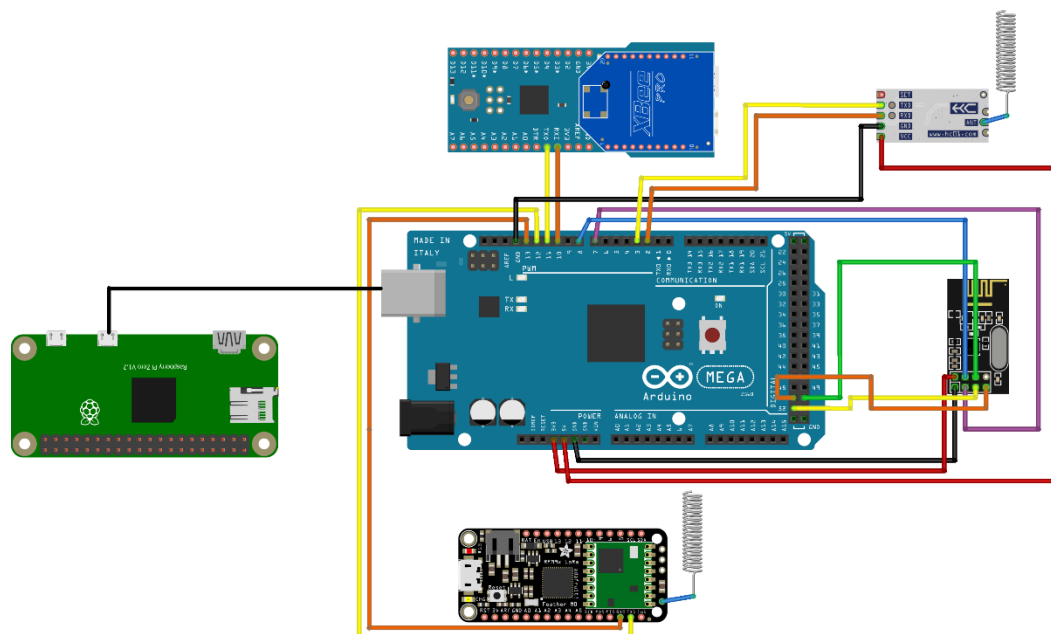


Rys. 4.16 - Schemat połączenia Sparkfun Fio v3 z modulem Xbee Pro oraz czujnikiem DHT22

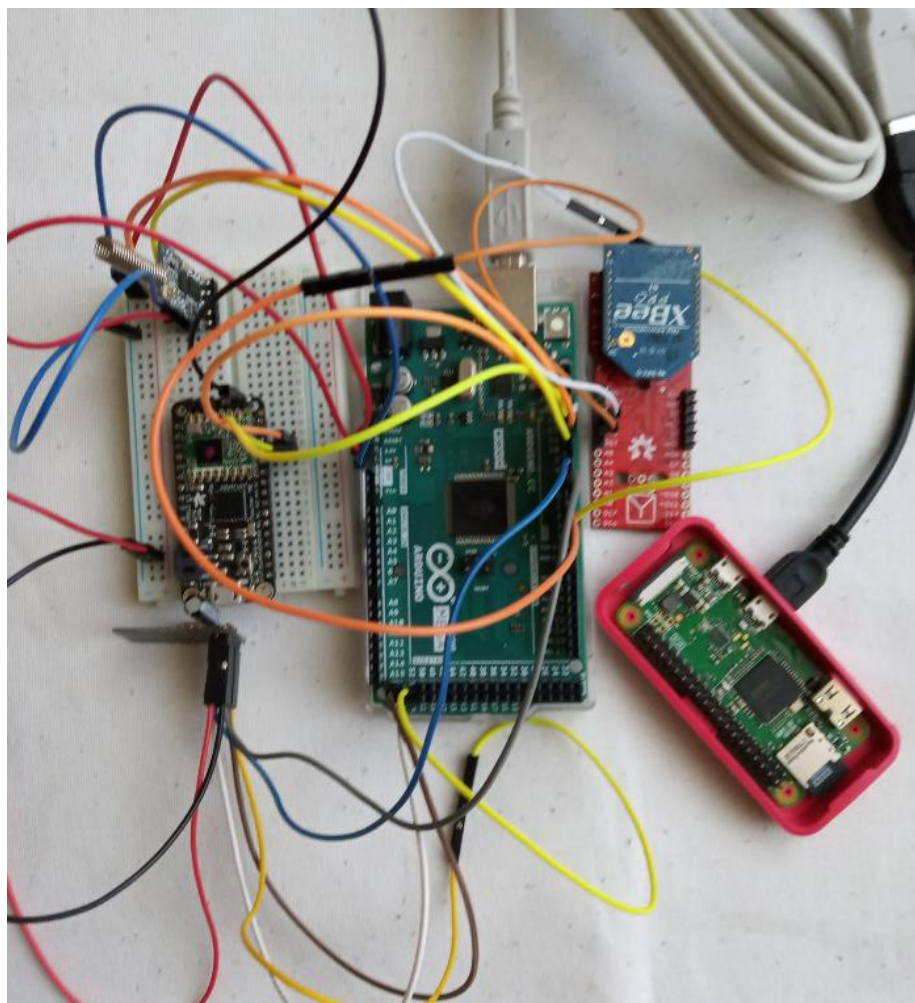


Rys. 4.17 - Połączenie Sparkfun Fio v3 z modułem Xbee oraz czujnikiem DHT22

Po zestawieniu urządzeń nadawczych połączono ze sobą moduły realizujące odbieranie danych i przekazywanie ich do systemu archiwizacji danych. Odbiorniki systemów bezprzewodowych HC-12, NRF24L01, Xbee i LoRa podłączono do Arduino mega 2560, która z kolei połączono poprzez interfejs szeregowy do Raspberry pi zero w (Rys. 4.18)(Rys. 4.19).



Rys. 4.18 - Schemat połączenie odbiorników z Arduino mega 2560 oraz Raspberry pi zero w



Rys. 4.19 - Połączenie odbiorników z Arduino mega 2560 oraz Raspberry pi zero w

4.4 Skrypty obsługujące moduły sensorowo-komunikacyjne

Zbieranie wartości temperatury z czujników oraz wysyłanie ich przez systemy komunikacji bezprzewodowej do bazy danych wymaga sporządzenia oraz wgrania na dedykowane urządzenia skryptów obsługujących te zadania. W tym celu napisany został kod programistyczny w języku Arduino dla mikrokontrolerów Arduino uno, Arduino mega 2560, Adafruit Feather M0 i Sparkfun fio v3 oraz w języku Python dla Raspberry pi zero w. Język Arduino oparty jest na popularnym języku do programowania systemów operacyjnych i zadań niskiego poziomu C. Python z kolei to język dynamicznie typowany ogólnego przeznaczenia, który według portal Stackoverflow był czwartym z kolei najpopularniejszym językiem programowania w 2019r.[29].

W niniejszym rozdziale opisane zostały najważniejsze fragmenty skryptów obsługujących urządzenia platformy dydaktycznej. Wszystkie skrypty wykorzystywane do obsługi systemów dostępne są w dodatku A.

4.4.1 HC-12

Skrypt obsługujący moduł HC-12 w pierwszym kroku pobiera wartość temperatury z czujnika DHT22. Następnie sprawdza czy otrzymane dane są poprawne i jeżeli tak to wysyła je do odbiornika HC-12 poprzez interfejs radiowy. Wszystko dzieje się w pętli co 10 min [Listing 1].

Listing 1 - Funkcja pobierająca wartość temperatury i wysyłająca je do odbiornika HC-12

```
void loop() {  
  
    float t = dht.readTemperature(); // Zczytanie wartości temperatury z czujnika  
  
    if (not (isnan(t))) { // Jeżeli dane są poprawne  
  
        char data[5] = t; // Stworzenie zmiennej char i przypisanie jej wartości temperatury  
  
        HC12.write(data); // Wysłanie danych  
  
    }  
  
    delay(600000); // Pauza na 10 min  
  
}
```

Odbiornik HC-12 czeka aż otrzyma dane, następnie tworzony jest identyfikator, który dodawany jest do otrzymanych danych i następnie cała informacja zapisywana jest w zmiennej, która zostanie wysłana do urządzenia Raspberry pi [Listing 2].

Listing 2 - Funkcja odbierająca dane z nadajnika HC-12

```
if (HC12.available()) { // Jeżeli interfejs HC12 jest dostępny  
  
    String HC12_output = HC12.readString(); // Zczytanie danych przesłanych od nadajnika HC-12  
  
    String HC12_id = "HC12"; // Utworzenie identyfikatora HC12  
  
    mess_for_rpi = mess_for_rpi + HC12_id + HC12_output; // Przypisanie danych do zmiennej  
  
}
```

4.4.2 NRF24L01

Todo: dokończyć opisywanie kodu (choć może z tego zrezygnuje)

5. System archiwizacji i wizualizacji danych

W celu przechowywania i wizualizacji danych temperatury wykorzystywanych w niniejszej pracy skorzystano z narzędzi umożliwiających wykonywanie tych czynności. Archiwizacja danych odbywa się w bazie danych InfluxDB. InfluxDB to darmowe oprogramowanie do przechowywania danych zbieranych w czasie, co czyni ją bardzo dobrym wyborem do przechowywania wartości temperatur pobieranych w ciągu pewnego zakresu czasu. Ponadto InfluxDB jest domyślnie wspieraną bazą przez wykorzystywane w tej pracy program do wizualizacji danych Grafana.



Rys. 5.1 - Logo baz danych InfluxDB stworzonej i utrzymywanej przez firmę InfluxData

Jak już wcześniej wspomniano graficzne przedstawienie danych odbywa się w programie Grafana. Grafana to wieloplatformowe darmowe narzędzie oparte na licencji open source. Rozwiązanie pozwala na pobieranie danych z licznych baz danych takich jak MySQL, PostgreSQL czy wykorzystywanych w tej pracy InfluxDB. Dzięki konfigurowalnym pulpitom nawigacyjnym użytkownik w wygodny sposób może monitorować dane zgromadzone w bazie danych.

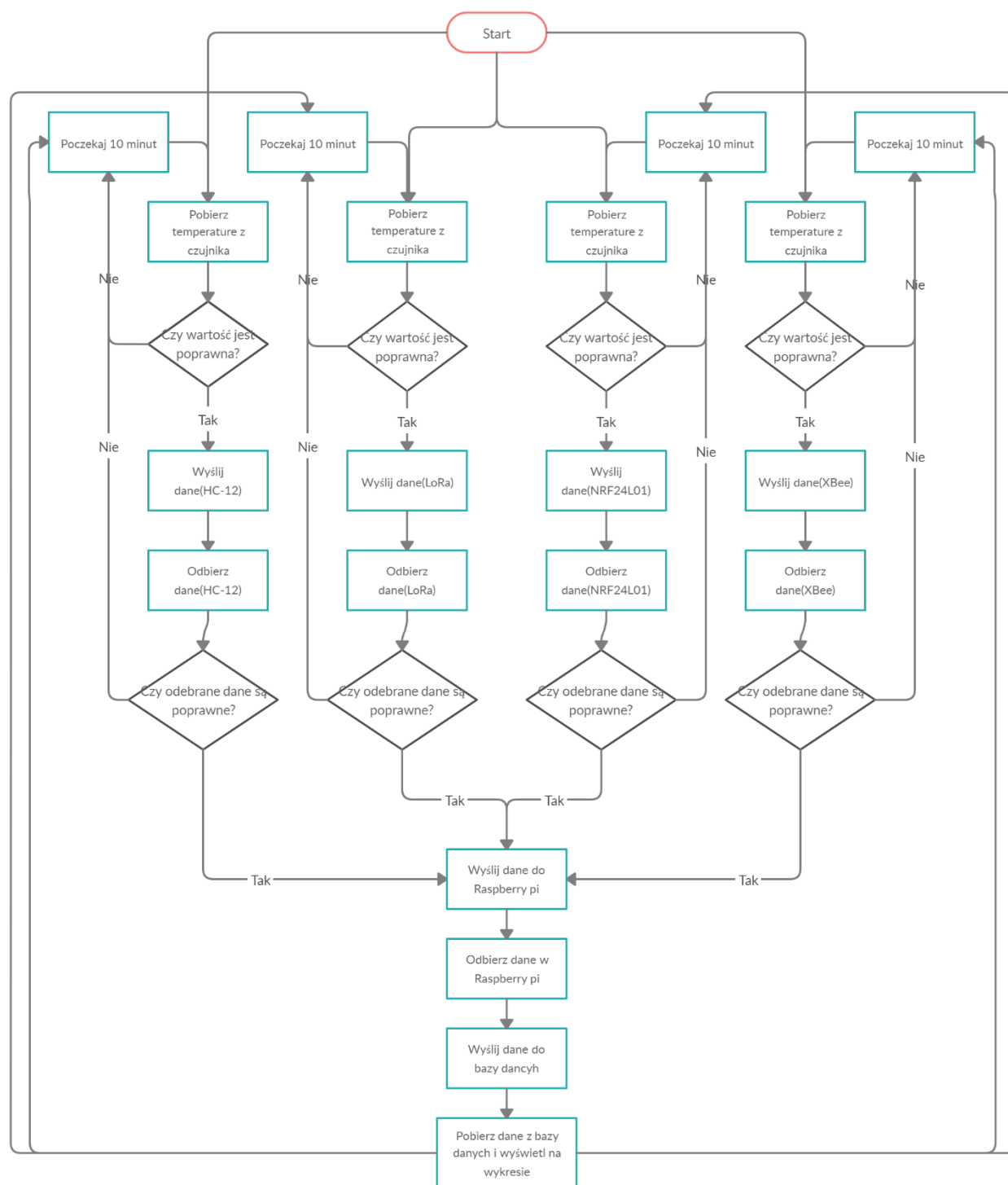


Rys. 5.2 - Grafana, przykładowy widok pulpitu z danymi

6. Maszyna stanów wielosystemowej platformy czujnikowo komunikacyjnej

Pierwszym krokiem działania platformy jest pobranie wartości temperatury z czujnika DHT-22, następnie sprawdzana jest poprawność pobranych danych poprzez weryfikację czy jest to wartości liczbowe typu float. Jeżeli dane są poprawne to nadajnik wysyła je do odbiornika poprzez interfejs radiowy każdej z technologii. Po odebraniu danych następuje ich ponowna weryfikacja podobnie jak poprzednio poprzez sprawdzenie czy są one wartością liczbową typu float. Po pomyślnej weryfikacji

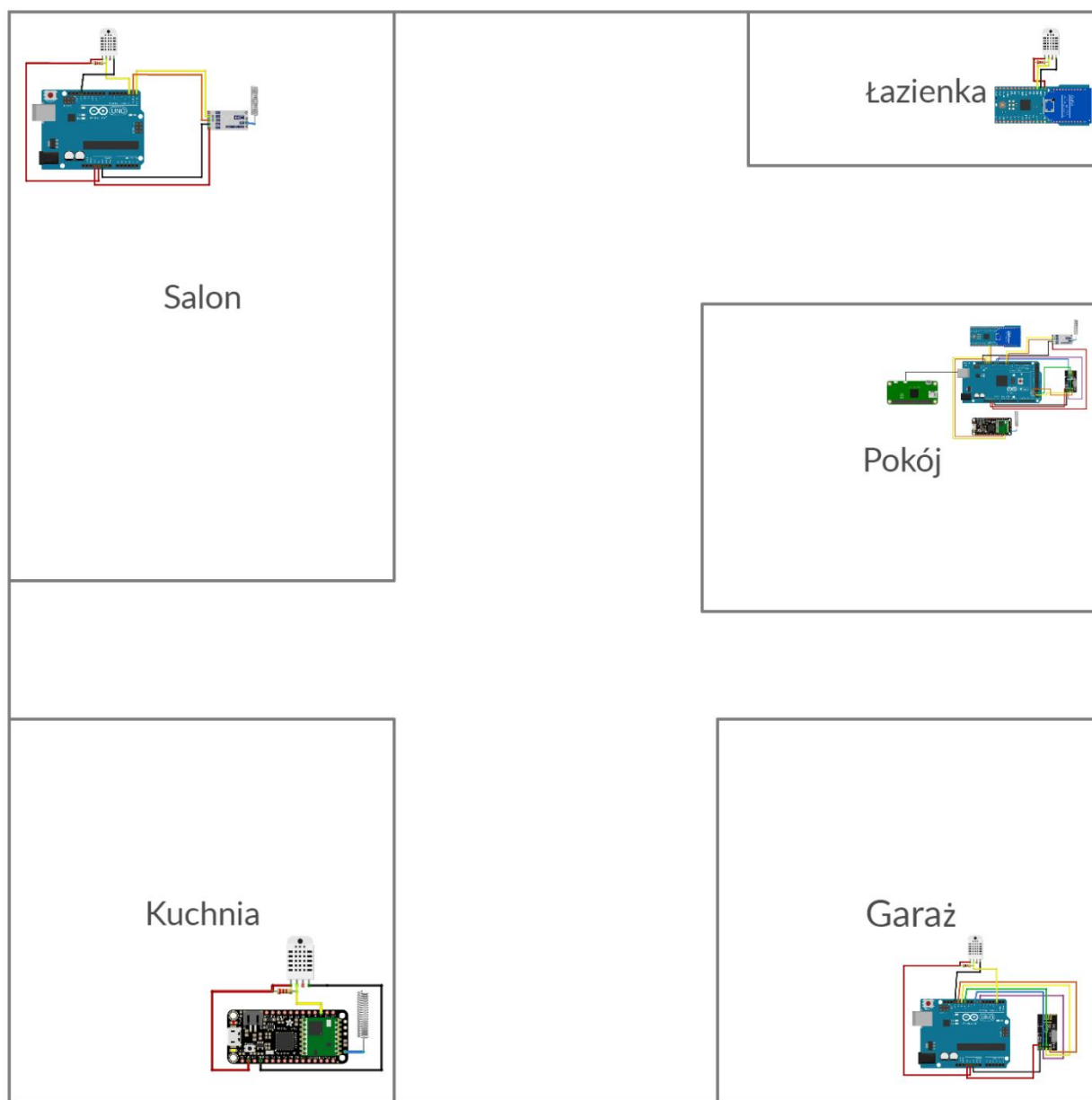
dane przekazywane do Raspberry pi a następnie do bazy danych. Ostatnim krokiem jest pobranie wartości temperatury z bazy danych przez program Grafana i wyświetlenie ich na graficznym wykresie. Cała operacja wykonywana jest w pętli co 10 minut (Rys. 6.1). Jeśli na którymś etapie weryfikacji dane uległy zniszczeniu procedura pomiaru wraca do początku. Każdy system bezprzewodowy działa niezależnie od innego co oznacza, że w przypadku utraty danych z np. modułu HC-12, dane z NRF24L01 zostaną przekazane do bazy danych i wyświetlone na wykresie.



Rys. 6.1 - Maszyna stanów wielosystemowej platformy czujnikowo komunikacyjnej

7. Przeprowadzenie pomiarów

Po pomyślnym zestawieniu każdego systemu bezprzewodowego przystąpiono do badań funkcjonalnych kompletnej platformy. Każdy zestaw pomiarowo transmisyjny ulokowany został w innym pomieszczeniu, gdzie pobierał temperaturę i następnie wysyłał ją do odbiornika. Schemat rozmieszczenia zestawów przedstawiony został na rysunku Rys. 7.1.



Rys. 7.1 - Schemat rozmieszczenia zestawów pomiarowych w budynku

Każdy z zestawów pomiarowych (HC-12 (salon), NRF24L01 (garaż), LoRa (kuchnia), XBee (łazienka)) został zaprogramowany by co 10 minut mierzyć temperaturę otoczenia w którym się znajduje, następnie dane te wysyłane są do odbiorników poszczególnych systemów, i w końcowej fazie pomiarów do systemu wizualizacji danych Grafana, która wyświetla zebrane wartości na wykresach graficznych. Badanie przeprowadzone zostało w godzinach od 02:00 do 09:00. Wyniki pomiarów ilustrują wykresy załączone na rysunku poniżej(Rys. 7.2).



Rys. 7.2 - Wyniki pomiaru temperatury w poszczególnych pomieszczeniach

Podczas przeprowadzania badań nie odnotowano sytuacji, w której dane były by przekłamane lub nie dotarły do odbiornika. Głównym czynnikiem wpływającym na poprawne działanie zestawów pomiarowych jest ich bliska odległość od odbiorników a zatem wysoki poziom odbieranego sygnału użytkowego.

8. Koncepcja ćwiczeń laboratoryjnych

W niniejszym rozdziale opracowana została koncepcji ćwiczeń laboratoryjnych z wykorzystaniem zbudowanej platformy. Todo: dopisać coś jeszcze

8.1 Cel ćwiczenia

Celem ćwiczenia jest:

- Zapoznanie się z technologiami przekazywania danych poprzez łącze bezprzewodowe we współczesnych sieciach IoT.
- Zapoznanie się z wybranymi platformami sprzętowymi wykorzystywanymi w sieciach sensorych
- Poznanie budowy oraz sposobów komunikacji wybranych modułów realizujących wymianę danych w sieciach bezprzewodowych
- Poznanie budowy i sposobu komunikacji z czujnikiem temperatury i wilgotności powietrza DHT22

8.2 Wprowadzenie

Todo: opisać wprowadzenie

8.3 Spis urządzeń

Do przeprowadzenia ćwiczenia niezbędne będą poniższe urządzenia:

- 2x Arduino Uno Rev3
- 1x Arduino Mega 2560

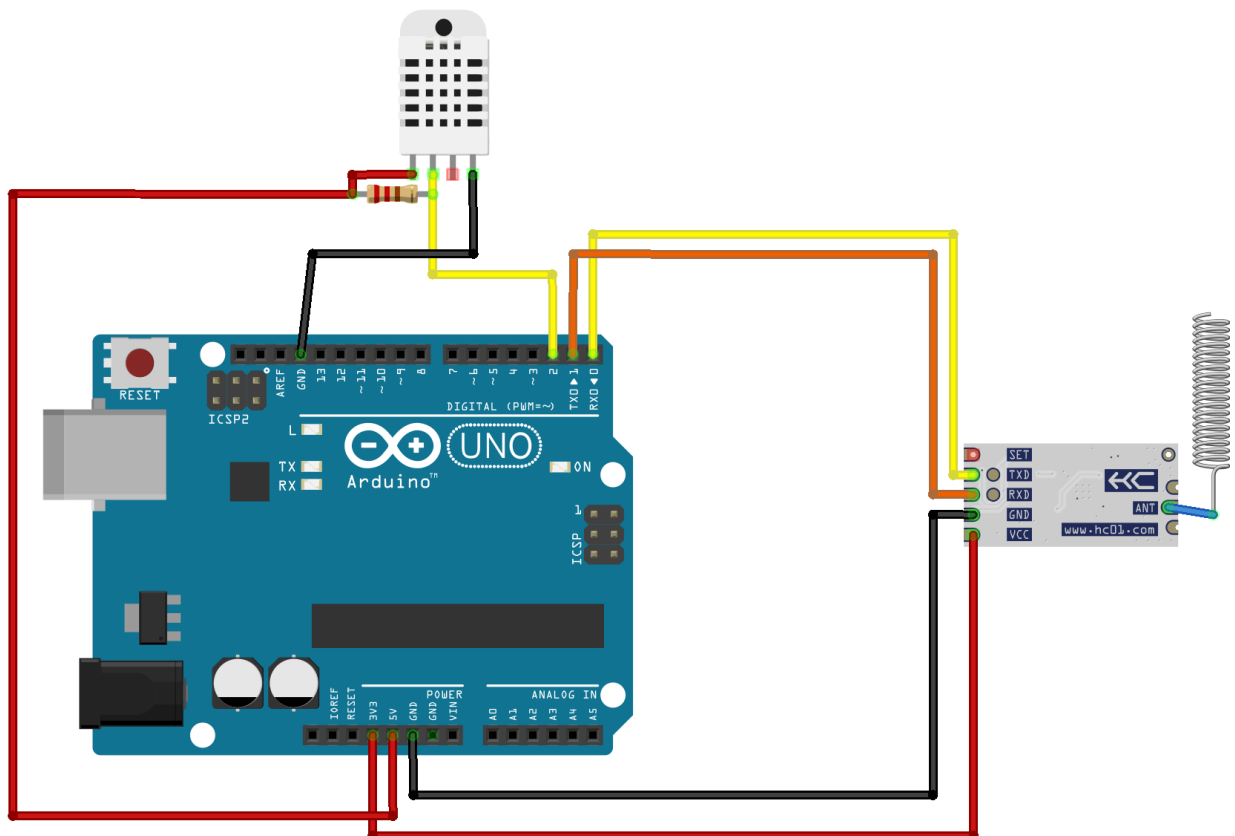
- 2x Adafruit Feather M0 z wbudowanym modułem Lora
- 2x SparkFun Fio v3
- 2x moduł HC-12
- 2x moduł NRF24L01
- 2x moduł XBee Pro S1
- 4x czujnik temperatury i wilgotności powietrza DHT22
- 4x rezystor o wartości 10KΩ

Dodatkowo potrzebne będą przewody przyłączeniowe do łączenia poszczególnych elementów platformy i płytki stykowe do komfortowego umiejscowienia urządzeń (opcjonalnie). Wymagane też będzie z oprogramowania Arduino oraz X-CTU do pisania skryptów dla platform sprzętowych.

8.4 Realizacja ćwiczenia

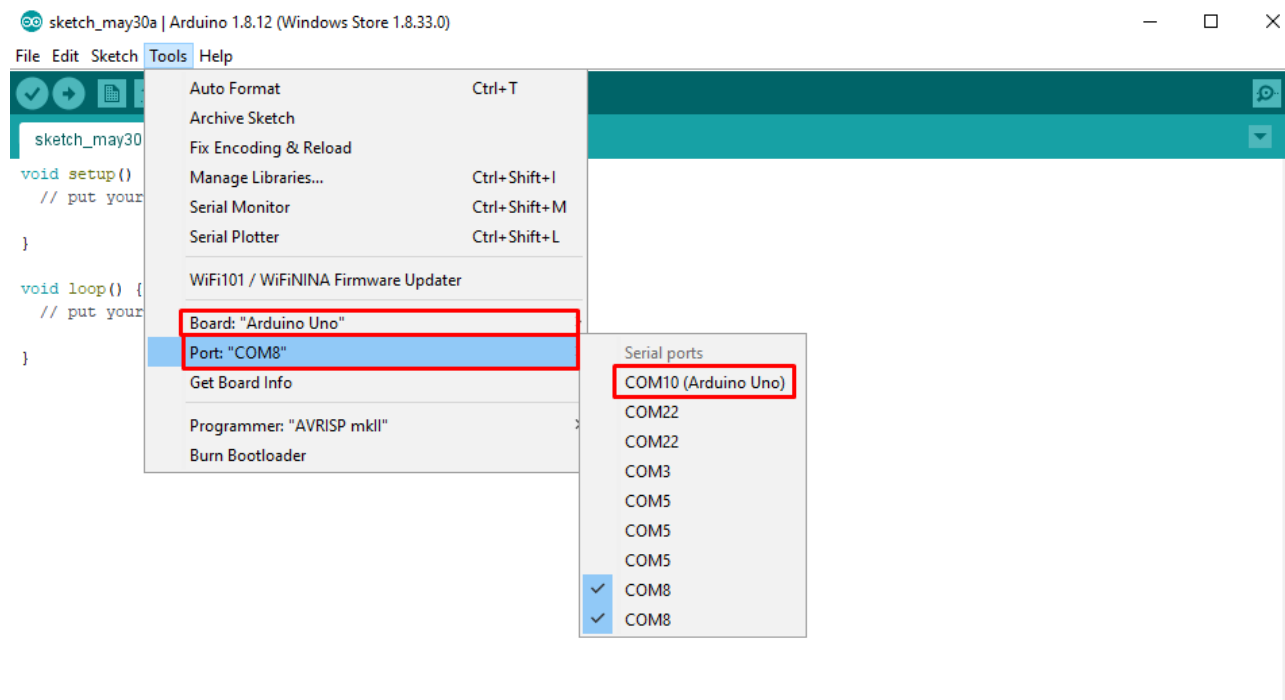
8.4.1 HC-12 nadajnik

Pierwszym etapem jest poprawne połączenie mikrokontrolera Arduino uno z modułem HC-12 oraz czujnikiem temperatury DHT22. Należy to zrobić w sposób przedstawiony na schemacie poniżej (Rys. 8). Dodatkowo płytkę Arduino należy podłączyć do komputera za pomocą kabla USB.



Rys. 8.1 - Schemat połączenia Arduino uno z modułem HC-12 i DHT22

Następnie należy włączyć program Arduino IDE w celu zaprogramowania płytki. Po uruchomieniu środowiska należy przejść do zakładki Tools → Port i wybrać port urządzenia com do którego wpięte jest Arduino. Również w zakładce Tools → Board proszę wybrać model Arduino Uno.

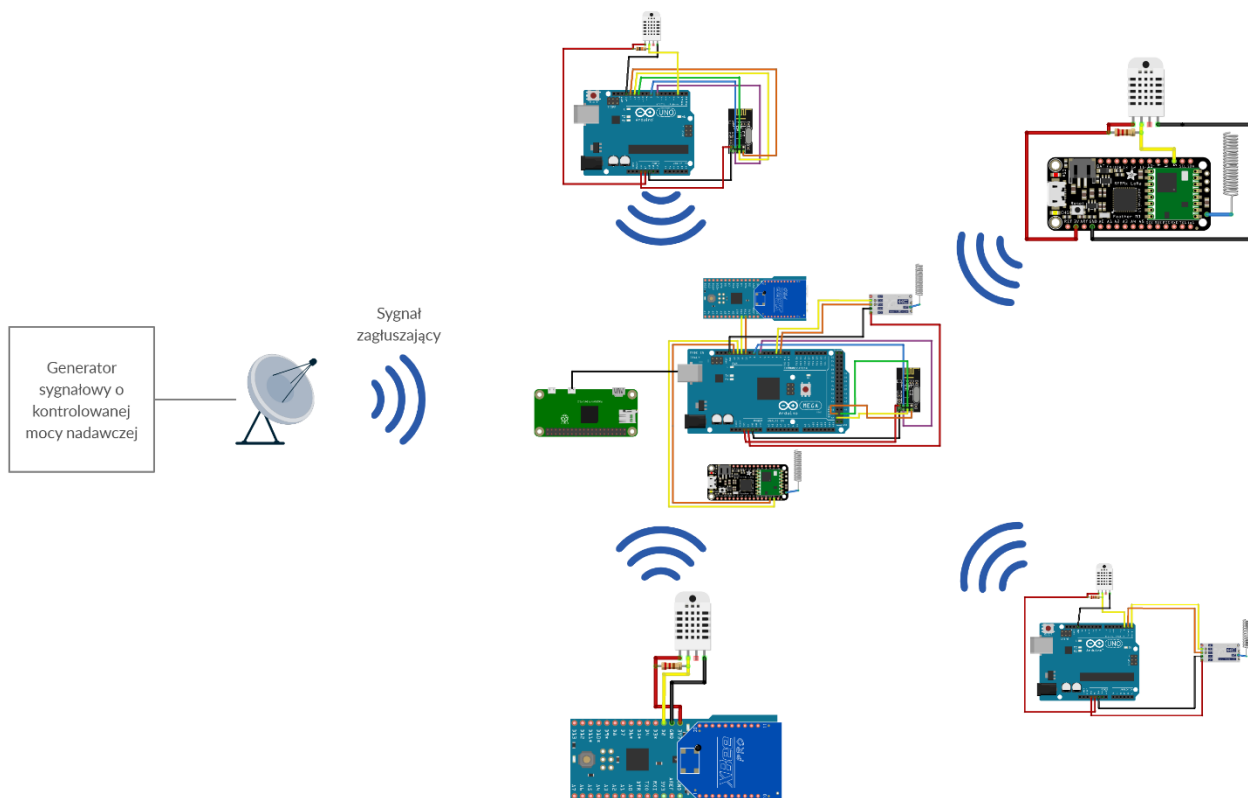


Rys. 8.2 - Wstępna konfiguracja środowiska programistycznego

Kolejnym krokiem jest wgranie odpowiedniego kodu na urządzenie Arduino. Skopiuj poniższy kod do programu Arduino IDE i kliknij przycisk Upload.

```
#include <SoftwareSerial.h>
#include "DHT.h"
#define DHTPIN 2 // Pin do którego podpięty jest DHT22
#define DHTTYPE DHT22 // Określenie typu czujnika DHT
DHT dht(DHTPIN, DHTTYPE); // Inicjalizacja instancji klasy DHT z numerem portu Arduino oraz typem czujnika
SoftwareSerial HC12(10, 11); // Inicjalizacja instancji klasy SoftwareSerial z numerami portów Arduino
void setup() {
    Serial.begin(9600);
    HC12.begin(9600); // Inicjalizacja komunikacji serialowej dla HC12
    dht.begin(); // Inicjalizacja komunikacji z DHT22
}
void loop() {
    float t = dht.readTemperature(); // Zczytanie wartości temperatury z czujnika
    Serial.println(t);
    if (not (isnan(t))) { // Jeżeli dane są poprawne
        Serial.println(t);
        char data[5] = {t}; // Stworzenie zmiennej char i przypisanie jej wartości temperatury
        HC12.write(data); // Wysłanie danych
    }
    delay(600000); // Pauza na 10 min
}
```

9. Propozycje dalszego rozwijania platformy



Rys. 9.1 - Koncepcja badania odporności platformy na zagłuszenia

10. Bibliografia

- [1] ITU-T Y.2060, „Overview of the Internet of things”
- [2] „The „Only” Coke Machine on the Internet”, Carnegie Mellon University, https://www.cs.cmu.edu/~coke/history_long.txt [ostatni dostęp: 01.05.2020r.]
- [3] Ashton Kevin, „That „Internet of Things” Thing”, <https://www.rfidjournal.com/that-internet-of-things-thing> [ostatni dostęp: 01.05.2020r.]
- [4] Evans Dave, „The Internet of Things How the Next Evolution of the Internet Is changing Everything”, https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf [ostatni dostęp: 01.05.2020r.]
- [5] Department of Information Technology, Krirk University, „Internet of Things: A review of applications & technologies”
- [6] System HomeKit, <https://developer.apple.com/homekit/> [ostatni dostęp: 05.05.2020r.]
- [7] Jason Baker, „6 open source home automation tools”, <https://opensource.com/tools/home-automation> [ostatni dostęp: 05.05.2020r.]
- [8] Demiris, G; Hensel, K, „Technologies for an aging society: a systematic review of "smart home" applications.”, IMIA Yearbook of Medical Informatics 2008
- [9] Aburukba, Raafat; Al-Ali, A. R.; Kandil, Nourhan; AbuDamis, Diala, „Configurable ZigBee-based control system for people with multiple disabilities in smart homes”
- [10] da Costa, CA; Pasluosta, CF; Eskofier, B; da Silva, DB; da Rosa Righi, „Internet of Health Things: Toward intelligent vital signs monitoring in hospital wards”, Artificial Intelligence in Medicine
- [11] “LoRa Modulation Basics”, Semtech Corporation, dostępny pod adresem: <https://web.archive.org/web/20190718200516/https://www.semtech.com/uploads/documents/an1200.22.pdf> [ostatni dostęp 13.05.2020r.]
- [12] Alireza Zourmand, Chan Wai Hung, Adrew Lai Kun Hing, Mohamman AbdulRegman, “Internet of Things (IoT) using LoRa technology”, dostępne pod adresem:

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8825008> [ostatni dostęp: 14.05.2020r.]

[13] "Grove LoRa radio user manual", https://wiki.seeedstudio.com/Grove_LoRa_Radio/ [ostatni dostęp: 15.05.2020r.]

[14] "DFRobot FireBeetle specification", https://wiki.dfrobot.com/FireBeetle_Covers-LoRa_Radio_433MHz_SKU_TEL0121 [ostatni dostęp: 14.05.2020r.]

[15] "HC-12 Wireless Serial Port Communication Module User Manual"

[16] "NRF24L01 Transceiver Product Specification", https://download.kamami.pl/p231723-nRF24L01P_Product_Specification_1_0.pdf [ostatni dostęp 15.05.2020r.]

[17] "IEEE Standard for Low-RateWireless Networks", <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8410916> [ostatni dostęp: 19.05.2020]

[18] C. Wang, T. Jiang, Q. Zhang, "Performance Analysis of the IEEE 802.15.4 Mac Layer", in Zigbee Network Protocols and Applications, CRS Press Taylor & Francis Group, 2014

[19] Abimael Escudero Osorio, Eddi A. García Sánchez, Gustavo Flores García, Sergio F. Hernández Machuca, Pablo S. Luna Lozano, "Evaluation of Wireless Network Based on ZigBee Technology Using XBee Modules", University of Veracruz, <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8920468> [ostatni dostęp: 19.05.2020r.]

[20] Phan Minh Linh An and Taehong Kim, "A Study of the Z-Wave Protocol: Implementing Your Own Smart Home Gateway", School of Information and Communication Engineering Chungbuk National University Cheongju, South Korea, <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8463281> [ostatni dostęp: 19.05.2020r.]

[21] Zhe Zheng, Lei Qiao, Liang Wang, "Discussion and Testing of 802.11ah Wireless Communication in Intelligent Substation", Research Department of Smart Chip, <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8403453> [ostatni dostęp: 19.05.2020r.]

[22] "LoRa RFM96W transceiver module description", https://github.com/TomaszBorusiewicz/praca_magisterska/blob/master/dokumentacje/RFM96W-V2.0.pdf [ostatni dostęp: 21.05.2020r.]

[23] "XBee Pro S1 User Manual", https://github.com/TomaszBorusiewicz/praca_magisterska/blob/master/dokumentacje/Xbee_user_manual.pdf [ostatni dostęp: 21.05.2020r.]

[24] Oficjalna dokumentacja urządzenia Arduino uno, <https://store.arduino.cc/arduino-uno-rev3> [ostatni dostęp: 21.05.2020]

[25] Oficjalna dokumentacja urządzenia Arduino Mega 2560, <https://store.arduino.cc/arduino-mega-2560-rev3> [ostatni dostęp: 21.05.2020]

[26] Oficjalna dokumentacja urządzenia SparkFun Fio v3, https://cdn.sparkfun.com/datasheets/Components/General%20IC/33244_SPCN.pdf [ostatni dostęp: 21.05.2020r.]

[27] Oficjalna dokumentacja urządzenia Raspberry pi zero w, <https://www.raspberrypi.org/products/raspberry-pi-zero-w/> [ostatni dostęp: 21.05.2020r.]

[28] "DHT22 sensor documentation", <https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf> [ostatni dostęp: 21.05.2020r.]

[29] "Developer Survey Results", <https://insights.stackoverflow.com/survey/2019#technology> [ostatni dostęp: 23.05.2020]

SPIS RYSUNKÓW

Rys. 3.1 - Mikrokontroler Feather M0 z modulem RFM9x LoRa firmy Adafruit	6
Rys. 3.2 - Moduł HC-12	7
Rys. 3.3 - Moduł NRF24L01	8
Rys. 3.4 - Moduł Xbee z anteną typu chip.	9
Rys. 3.5 - Moduł Xbee z anteną typu wire.	9
Rys. 3.6 - Moduł Xbee z anteną typu u.FL.	10
Rys. 3.7 - Moduł Xbee z anteną typu RPSMA.	10
Rys. 3.8 - Moduł Xbee z anteną typu Trace (PCB).	11
Rys. 3.9 - Czujnik ruchu, temperatury i natężenia światła z modulem komunikacyjnym Z-Wave.	12
Rys. 3.10 - Zalety technologii WiFi HaLow [źródło: https://www.wi-fi.org/discover-wi-fi/wi-fi-halow]	12
Rys. 3.11 - Moduł bezprzewodowy kompatybilny z 802.11ah HaLow [źródło: https://www.silextechnology.com/unwired/industry-first-802.11ah-halow-wireless-module-for-iot-devices]	13
Rys. 3.12 - Architektura sieci Bluetooth.....	14
Rys. 4.1 - Moduł HC-12	15
Rys. 4.2 - Moduł NRF24L01	16
Rys. 4.3 - Mikrokontroler Adafruit Feather M0 z modulem radiowym LoRa	17
Rys. 4.4 - Moduł Xbee Pro S1	18
Rys. 4.5 - Mikrokontroler Arduino Uno	19
Rys. 4.6 - Mikrokontroler Arduino Mega 2560	20
Rys. 4.7 - Mikrokontroler SparkFun Fio v3	21
Rys. 4.8 - Mikrokontroler Raspberry pi zero w	22
Rys. 4.9 - Czujnik temperatury i wilgotności powietrza DHT22	23
Rys. 4.10 - Schemat połączenia Arduino uno z modulem HC-12 oraz czujnikiem DHT22	24
Rys. 4.11 - Połączenie Arduino uno z modulem HC-12 oraz czujnikiem DHT22	24
Rys. 4.12 - Schemat połączenia Arduino uno z modulem NRF24L01 oraz czujnikiem DHT22	25
Rys. 4.13 - Połączenie Arduino uno z modulem NRF24L01 oraz czujnikiem DHT22	26
Rys. 4.14 - Schemat połączenia Adafruit Feather M0 z czujnikiem DHT22	26
Rys. 4.15 - Połączenie Adafruit Feather M0 z czujnikiem DHT22.....	27
Rys. 4.16 - Schemat połączenia Sparkfun Fio v3 z modulem Xbee Pro oraz czujnikiem DHT22	27
Rys. 4.17 - Połączenie Sparkfun Fio v3 z modulem Xbee oraz czujnikiem DHT22	28
Rys. 4.18 - Schemat połączenia odbiorników z Arduino mega 2560 oraz Raspberry pi zero w	28
Rys. 4.19 - Połączenie odbiorników z Arduino mega 2560 oraz Raspberry pi zero w	29
Rys. 5.1 - Logo baz danych InfluxDB stworzonej i utrzymywanej przez firmę InfluxData.....	31
Rys. 5.2 - Grafana, przykładowy widok pulpitu z danymi.....	31
Rys. 6.1 - Maszyna stanów wielosystemowej platformy czujnikowo komunikacyjnej	32
Rys. 7.1 - Schemat rozmieszczenia zestawów pomiarowych w budynku	33
Rys. 7.2 - Wyniki pomiaru temperatury w poszczególnych pomieszczeniach.....	34
Rys. 9.1 - Koncepcja badania odporności platformy na zagłuszenia.....	37

Dodatek A. Kody źródłowe urządzeń Arduino uno, Arduino mega, Adafruit Feather M0, Sparkfun fio v3 oraz Raspberry pi zero w

Arduino uno + moduł HC-12 nadajnik:

```
#include <SoftwareSerial.h>

#include "DHT.h"

#define DHTPIN 2 // Pin do którego podpięty jest DHT22

#define DHTTYPE DHT22 // Określenie typu czujnika DHT

DHT dht(DHTPIN, DHTTYPE); // Inicjalizacja instancji klasy DHT z numerem portu Arduino oraz typem czujnika

SoftwareSerial HC12(10, 11); // Inicjalizacja instancji klasy SoftwareSerial z numerami portów Arduino

void setup() {

    HC12.begin(9600); // Inicjalizacja komunikacji serialowej dla HC12

    dht.begin(); // Inicjalizacja komunikacji z DHT22

}

void loop() {

    float t = dht.readTemperature(); // Zczytanie wartość temperatury z czujnika

    if (not (isnan(t))){ // Jeżeli dane są poprawne

        char data[5] = t; // Stworzenie zmiennej char i przypisanie jej wartości temperatury

        HC12.write(data); // Wysłanie danych

    }

    delay(600000); // Pauza na 10 min

}
```

Arduino uno + NRF24L01 nadajnik:

```
#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#include "DHT.h"

#define DHTPIN 2 // Pin do którego podpięty jest DHT22

#define DHTTYPE DHT22 // Określenie typu czujnika DHT

RF24 radio(7, 8); // Inicjalizacja instancji klasy RF24 z numerami pinów CNS oraz CE

const byte address[6] = "00001"; // Inicjalizacja zmiennej address dzięki której urządzenia NRF24L01 będą mogły
komunikować się między sobą

DHT dht(DHTPIN, DHTTYPE); // Inicjalizacja instancji klasy DHT

void setup() {

    dht.begin(); // Inicjalizacja komunikacji z DHT22

    radio.begin(); // Inicjalizacja obiektu radia RF24

    radio.openWritingPipe(address); // Przypisanie adresu przez który wysyłana będzie wiadomość

    radio.setPALevel(RF24_PA_MIN); // Ograniczenie mocy nadajnika do minimum

    radio.stopListening(); // Wyłączenie nasłuchiwanie

}

void loop() {

    float t = dht.readTemperature(); // Zczytanie wartości temperatury z czujnika

    if (not (isnan(t))){ // Jeżeli dane są poprawne

        char data[5] = {t}; // Stworzenie zmiennej char i przypisanie jej wartości temperatury

        radio.write(&data, sizeof(data)); // Wysłanie danych

    }

    delay(600000); // Pauza na 10 min

}
```

Adafruit Feather MO + LoRa nadajnik

```
#include <SPI.h>

#include <RH_RF95.h>

#include "DHT.h"

#define DHTPIN 12 // Pin do którego podpięty jest DHT22

#define DHTTYPE DHT22 // Określenie typu czujnika DHT

DHT dht(DHTPIN, DHTTYPE); // Inicjalizacja instancji klasy DHT z numerem portu Arduino oraz typem czujnika

#define RFM95_CS 8 // Numer pinu CS

#define RFM95_RST 4 // Numer pinu RST

#define RFM95_INT 3 // Numer pinu INT

#define RF95_FREQ 434.0 // Określenie częstotliwości na której pracuje radio Lor

RH_RF95 rf95(RFM95_CS, RFM95_INT); // Inicjalizacja instancji klasy rf95

void setup()
{
    dht.begin(); // Inicjalizacja komunikacji z DHT22

    pinMode(RFM95_RST, OUTPUT); // Ustawienie pinu RST jako wyjściowy
    digitalWrite(RFM95_RST, HIGH); // Przypisanie stanu wysokiego pinowi RST

    Serial.begin(9600); // Inicjalizacja komunikacji serialowej do Arduino mega 2560

    rf95.setTxPower(5, false); // Ustawienie mocy nadawania na 5dBm
}

void loop()
{
    int t = dht.readTemperature(); // Zczytanie wartości temperatury z czujnika

    if (not (isnan(t))) { // Jeżeli dane są poprawne

        char data[5] {t}; // Stworzenie zmiennej char i przypisanie jej wartości temperatury

        rf95.send((uint8_t *)data, 5); // Wysłanie danych

        rf95.waitPacketSent(); // Oczekiwanie na poprawne wysłanie pakietu

    }

    delay(600000); // Pauza na 10 min
}
```

Adafruit Feather M0 + LoRa odbiornik

```
#include <SPI.h>

#include <RH_RF95.h>

#define RFM95_CS 8 // Numer pinu CS

#define RFM95_RST 4 // Numer pinu RST

#define RFM95_INT 3 // Numer pinu INT

#define RF95_FREQ 434.0 // Określenie częstotliwości na której pracuje radio Lor

RH_RF95 rf95(RFM95_CS, RFM95_INT); // Inicjalizacja instancji klasy rf95

void setup()
{
    pinMode(RFM95_RST, OUTPUT); // Ustawienie pinu RST jako wyjściowy
    digitalWrite(RFM95_RST, HIGH); // Przypisanie stanu wysokiego pinowi RST
    Serial.begin(9600); // Inicjalizacja komunikacji serialowej do Arduino mega 2560
}

void loop()
{
    if (rf95.available()) // Jeżeli interfejs radiowy jest dostępny
    {
        uint8_t buf[RH_RF95_MAX_MESSAGE_LEN]; // Utworzenie zmiennej na otrzymane dane
        uint8_t len = sizeof(buf); // Obliczenie długości zmiennej na dane
        if (rf95.recv(buf, &len)) // Jeżeli odbiornik otrzymał dane
        {
            Serial.println((char*)buf); // Wyślij dane do urządzenia Arduino mega 2560
        }
    }
}
```

Sparkfun Fio v3 + XBee nadajnik

```
#include "DHT.h"

#define DHTPIN 2 // Pin do którego podpięty jest DHT22
#define DHTTYPE DHT22 // Określenie typu czujnika DHT

DHT dht(DHTPIN, DHTTYPE); // Inicjalizacja instancji klasy DHT

void setup() {
  Serial.begin(9600); // Inicjalizacja komunikacji serialowej Arduino mega 2560
  dht.begin(); // Inicjalizacja komunikacji z DHT22
}

void loop() {
  float t = dht.readTemperature(); // Zczytanie wartość temperatury z czujnika
  if (not (isnan(t))) { // Jeżeli dane są poprawne
    char data[5] = {t}; // Stworzenie zmiennej char i przypisanie jej wartości temperatury
    Serial.write(data); // Wysłanie danych
  }
  delay(600000); // Pauza na 10 min
}
```

Sparkfun Fio v3 + XBee odbiornik

```
void setup() {
  Serial.begin(9600); // Inicjalizacja komunikacji serialowej Arduino mega 2560
}

void loop() {
  if (Serial.available() > 0) { // Jeżeli interfejs serialowy jest dostępny
    read_data = Serial.read(); // Zczytaj dane otrzymane od nadajnik
    Serial.println(read_data); // Wyślij dane do Arduino mega 2560
  }
}
```

Arduino mega 2560:

```
#include <SoftwareSerial.h>

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

// ##### HC-12 #####
SoftwareSerial HC12(10, 11); // Inicjalizacja instancji klasy serialowej dla HC12
// ##### HC-12 #####
// ##### LORA #####
SoftwareSerial LORA(12, 13); // Inicjalizacja instancji klasy serialowej dla LoRy
// ##### LORA #####
// ##### NRF24L01 #####
RF24 radio(7, 8); // Inicjalizacja instancji klasy nRF24L01 z numerami pinów CNS oraz CE

const byte address[6] = "00001"; // Inicjalizacja zmiennej address dzięki której urządzenia NRF24L01 będą mogły
komunikować się między sobą

// ##### NRF24L01 #####
// ##### Xbee #####
SoftwareSerial Xbee(4, 5); // Inicjalizacja instancji klasy serialowej dla Xbee
// ##### Xbee #####

void setup() {

  Serial.begin(9600); // Inicjalizacja komunikacji serialowej do Raspberry pi
  HC12.begin(9600); // Inicjalizacja komunikacji serialowej dla HC12
  LORA.begin(9600); // Inicjalizacja komunikacji serialowej dla Lory
  Xbee.begin(9600); // Inicjalizacja komunikacji serialowej dla Xbee
  radio.begin(); // Inicjalizacja obiektu radia RF24

  radio.openReadingPipe(0, address); // Przypisanie adresu przez który odbierana będzie wiadomość
  radio.setPALevel(RF24_PA_MIN); // Ograniczenie mocy nadajnika do minimum
  radio.startListening(); // Uruchomienie nasłuchiwania
}
```



```

void loop() {
    String mess_for_rpi;

    if (HC12.available()) { // Jeżeli interfejs HC12 jest dostępny

        String HC12_output = HC12.readString(); // Zczytanie danych przesłanych od nadajnika HC-12

        String HC12_id = "HC12"; // Utworzenie identyfikatora HC12

        mess_for_rpi = mess_for_rpi + HC12_id + HC12_output; // Przypisane danych do zmiennej
    }

    if (LORA.available()) { // Jeżeli interfejs HC12 jest dostępny

        String LORA_output = LORA.readString(); // Zczytanie danych przesłanych od nadajnika LoRa

        String LORA_id = "LORA"; // Utworzenie identyfikatora Lora

        mess_for_rpi = mess_for_rpi + LORA_id + LORA_output; // Przypisane danych do zmiennej
    }

    if (radio.available()) { // Jeżeli interfejs nRF24L01 jest dostępny

        char text[32] = ""; // Utworzenie zmiennej do której zapisywane są dane z nRF24L01

        nRF24L01_output = radio.readString(&text, sizeof(text)); // Zczytanie danych przesłanych od nadajnika
nRF24L01

        String NRF24L01_id = "NRF24L01"; // Utworzenie identyfikatora nRF24L01

        mess_for_rpi = mess_for_rpi + NRF24L01_id + nRF24L01_output; // Przypisane danych do zmiennej
    }

    if (Xbee.available()){ // Jeżeli interfejs Xbee jest dostępny

        Xbee_output = Xbee.readString(); // Zczytanie danych przesłanych od nadajnika Xbee

        String Xbee_id = "XBEE"; // Utworzenie identyfikatora Xbee

        mess_for_rpi = mess_for_rpi + Xbee_id + Xbee_output; // Przypisane danych do zmiennej
    }

    if (mess_for_rpi.length() > 0){ // Jeżeli zmienna posiada dane z jakiegoś nadajnika

        Serial.println(mess_for_rpi); // Wysłane danych do raspberry pi
    }

}

```

Raspberry pi zero w:

```
from re import search
```

```
from serial import Serial
```

```
from subprocess import run
```

```
from time import sleep
```

```
def main():
```

```
    data = get_data_from_serial()
```

```
    formatted_data = format_data(data)
```

```
    send_data(formatted_data)
```

```
def send_data(data):
```

```
    if data:
```

```
        for k, v in data.items():
```

```
            run(["curl", "-i", "-XPOST", "http://40.114.77.143:8086/write?db=dane", "--data-binary",
```

```
                "{}", host=admin value={}".format(k, v)], shell=False)
```

```
            sleep(2)
```

```
def get_data_from_serial():
```

```
    s = Serial("/dev/ttyACM0", 9600, timeout=1)
```

```
    s.flush()
```

```
    data = s.readline().decode("utf-8").rstrip()
```

```
    if data != "":
```

```
        return data
```

```

def format_data(data):
    formatted_data = dict()

    if "HC12" in data:
        hc_12_data = search("(?<=HC12).....", data).group()
        formatted_data["salon"] = hc_12_data

    if "LORA" in data:
        lora_data = search("(?<=LORA).....", data).group()
        formatted_data["kuchnia"] = lora_data

    if "NRF24L01" in data:
        nrf24l01_data = search("(?<=NRF24L01).....", data).group()
        formatted_data["garaz"] = nrf24l01_data

    if "XBEE" in data:
        xbee_data = search("(?<=XBEE).....", data).group()
        formatted_data["lazienka"] = xbee_data

    return formatted_data

if __name__ == "__main__":
    counter = 0

    while True:
        try:
            main()
        except:
            counter += 1

            if counter <= 10:
                continue
            else:
                run(["sudo", "reboot"])

```