BIBLIOTEKA
POLSKIEGO KRÓTKOFALOWCA

33

Amatorska telemetria

Krzysztof Dąbrowski
OE1KDA

Wiedeń 2017

Amatorska telemetria

Krzysztof Dąbrowski OE1KDA

Wydanie 1
Wiedeń, czerwiec 2017

Spis treści

Sommaire
Télémétrie amateur

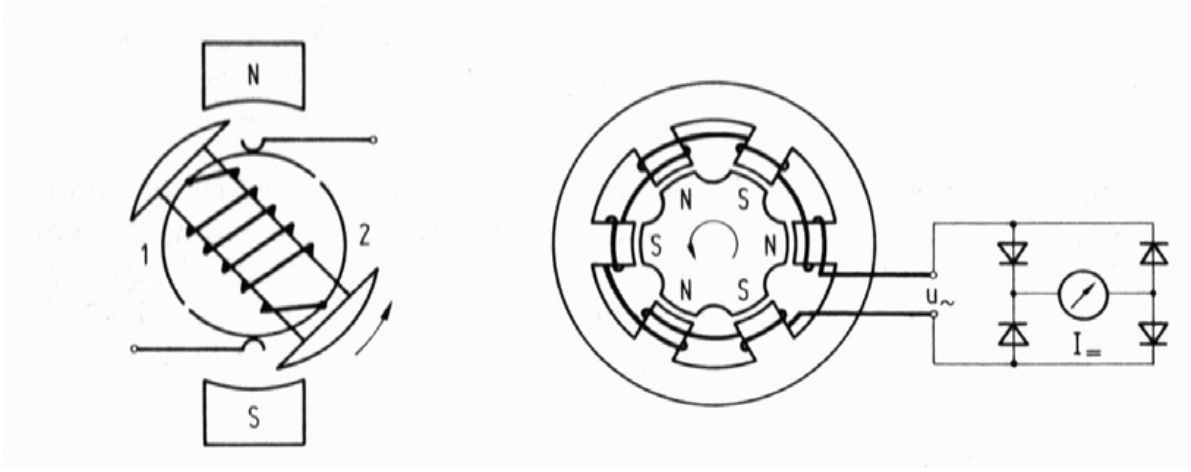
Wstęp

Do najczęściej wymienianych informacji w łącznościach amatorskich należą raporty słyszalności, imiona, lokalizacje stacji, krótkie opisy wyposażenia, czasami rozmowy schodzą na tematy meteorologiczne albo techniczne. W trakcie zawodów lub łączności DX-owych wymieniane jest jedynie niezbędne do zaliczenia łączności minimum danych. Automatycznie pracujące radiolatarnie ograniczają się przeważnie do nadawania znaku i tylko nieliczne dodają do nich dłuższe komunikaty o warunkach propagacji lub innej treści.
Od czasu rozpowszechnienia się systemu APRS w komunikatach podawane są nie tylko dokładne współrzędne geograficzne ale również i różnego rodzaju dane pomiarowe: napięcia zasilania urządzeń, temperatury panujące wewnątrz obudowy, informacje meteorologiczne itp.
Ogólnie rzecz biorąc możliwe jest rozpowszechnianie lub wymiana wszelkiego rodzaju danych pomiarowych, które mogą zainteresować szersze grono odbiorców. Mogą one być związane z wszelkiego rodzaju eksperymentami i obserwacjami prywatnymi, szkolnymi, studenckimi i temu podobnymi.
Skrypt niniejszy poświęcony jest różnorodnym możliwościom i konceptom amatorskich pomiarów wielkości meteorologicznych, technicznych czy środowiskowych mogących interesować nie tylko krótkofalowców. Część z nich mogłaby stanowić temat dodatkowych zajęć szkolnych i pozaszkolnych dla młodzieży przyczyniając się do zainteresowania jej problematyką ochrony środowiska itp., zasadami pomiarów, metodami pomiarowymi i związaną z tym techniką.
Otrzymane dane mogą być transmitowane w pasmach amatorskich jeżeli nauczyciel jest licencjonowanym krótkofalowcem albo w pasmach ogólnie dostępnych CB czy 868 MHz w innych przypadkach. Mogą one być także przekazywane za pośrednictwem łączy świetlnych lub w inny nie wymagajacy uzyskania szczególnych zezwoleń sposób.
W skrypcie przedstawione są zasady pomiarów, przykłady konstrukcji własnych i rozwiązań fabrycznych różnego rodzaju czujników i układów pomiarowych, ich wykorzystanie we współpracy z mikrokomputerami (głównie na przykładzie „Arduino” i częściowo także „Maliny”) i niektóre standardy transmisji danych, w tym APRS, „LoRa” i Xbee. Oczywiście nic nie stoi na przeszkodzie w wykorzystaniu innych zbliżonych układów mikrokomputerów, a nawet konstrukcji własnych opartych o mikroprocesory PIC lub AVR.
Przytoczone przykłady najprostszych programów mają pomóc czytelnikom zainteresowanym programowaniem w tworzeniu własnych programów dostosowanych do ich potrzeb, konceptów i zainteresowań. Może to okazać się również przydatne dla zainteresowanej nauką programowania młodzieży, dla której „Arduino” i „Malina” są świetnymi platformami dydaktycznymi.
Zaprezentowane w skrypcie pomysły i rozwiązania mają stanowić w zamyśle autora nie tylko gotowe do odwzorowania przykłady ale również inspirację do tworzenia własnych, także bardziej rozbudowanych rozwiązań. Kiomentarze wprzytoczonych jako przykłady programach zostały przetlumaczone przez autora ale nazwy zmiennych pozostawiono i inne szczegóły w oryginale.
Podstawowe informacje o mikrokomputerach „Arduino”, „Malina” i podobnych oraz o ich programowaniu można znaleźć w literaturze dostepnej m.in. w sklepie internetowym wydawnictwa AVT [2], w działach technicznych lepszych księgarni oraz w tomach 20, 21 i 24 „Biblioteki polskiego krótkofalowca”. Tematyka APRS i DPRS jest szczegółowo poruszona w tomie 8 tej serii.
Przedstawione rozwiązania mają charakter eksperymentalny i dydaktyczny i nie powinny być w żadnym wypadku stosowane tam, gdzie od niezawodności i dokładności sprzętu zależy bezpieczeństwo ludzi lub mienia.

Krzysztof Dąbrowski OE1KDA
Wiedeń
8 czerwca 2017

Pomiary i metody

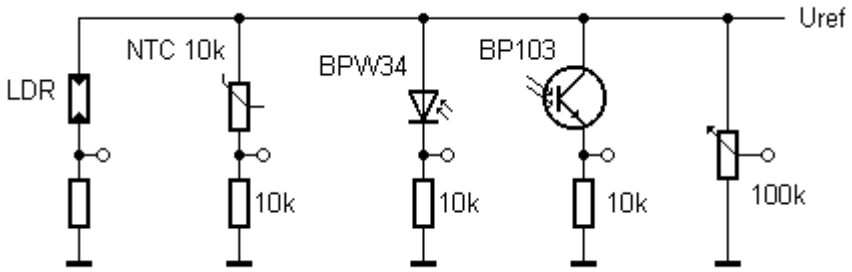
Pomiar wartości nieelektrycznych j.np. temperatury, ciśnienia, wilgotności lub oświetlenia wymaga zastosowania odpowiedniego czujnika przetwarzającego tą wielkość na wielkości elektryczne: napięcie, prąd, częstotliwość sygnału lub na postać dwójkową przeznaczoną do przetwarzania komputerowego. Do najważniejszych parametrów czujników należą czułość (stopień wrażliwości na mierzone efekty zjawisk), selektywność (stopień wrażliwości na inne efekty i zjawiska – w tym przypadku niepożądane) i stabilność (stałość parametrów i ich niezależność od czynników zewnętrznych i czasu).
Oprócz czujników zwykłych – elementów wrażliwych na określone wielkości i włączonych do koniecznych układów elektronicznych stosowane są czujniki inteligentne wyposażone we własny mikroprocesor. Są to też często urządzenia wielokanałowe mierzące różne wielkości np. ciśnienie i temperaturę lub ciśnienie, temperaturę i wilgotność lub te same wielkości ale w różnych miejscach (punktach). Inteligentne czujniki dostarczają przeważnie danych w postaci wygodnej do dalszego przetwarzania komputerowego i są skalibrowane tak, aby dostarczać danych z potrzebną dokładnością.
Zamiana wielkości nieelektrycznych na elektryczne może polegać na zmianie oporności elementu (pod wpływem rozciągania, nacisku, temperatury), występowaniu efektu piezoelektrycznego (pod wpływem nacisku, siły itp.), występowaniu efektu Halla (zmiany oponości pod wpływem zewnętrznego pola magnetycznego), zmianie indukcyjności cewki pod wpływem czynników zewnętrznych (ruchu rdzenia powiązanego z pływakiem itp.) albo zmianie pojemności w wyniku ruchu okładek kondensatora albo zmiany właściwości jego dielektryka (przykładowo pod wpływem wilgotności, stopnia zanurzenia elektrod w płynie nieprzewodzącym). W systemach optycznych używane są tarcze lub inne ruchome elementy podzielone na segmenty przezroczyste i nieprzezroczyste lub odbijajace i nie odbijające światło. Ruch elementu może powodować generację impulsów albo kodów dwójkowych i podobnych. Pomiarów odległości i szybkości można dokonywać także korzystając z odbić impulsów świetlnych (laserowych), radarowych lub ultradźwiękowych. W pomiarach szybkości obrotowej stosowane są tachometry – silniki elektryczne pracujące jako generatory. Mogą to być zarówno generatory prądu stałego jak i zmiennego. Do bardziej skomplikowanych systemów należą rozwiązania korzystające z kamer i analizujace komputerowo otrzymane obrazy.
W zależności od zasady działania zależność między wielkością mierzoną i wielkością elektryczną może mieć charakter liniowy lub nie. Czujniki zależnie od ich charakteru mogą być włączane w układy dzielników oporowych albo pojemnościowych, w układy mostkowe, występować w charakterze elementów obwodów drgających generatorów itp.



Rys. 1.1.1. Tachometr prądu stałego (po lewej) i zmiennego (po prawej)

Dla obróbki komputerowej otrzymane na zaciskach wyjściowych układu pomiarowego analogowe wielkości elektryczne muszą zostać przetworzone na postać cyfrową za pomocą przetwornika analogowo-cyfrowego (a-c). „Arduino” i szereg modeli mikroprocesorów z serii PIC posiadają wbudowane przetworniki a-c. Przeważnie są to przetworniki 8- lub 10-bitowe. „Malina” i podobne mikrokomputery wymagają natomiast podłączenia zewnętrznego przetwornika. Szczególnie interesujące do tego celu są przetworniki wyposażone w złącze I2C, *1-Wire* lub podobne. W przypadku korzystania z zewnętrznego przetwornika konstruktorzy mają więcej swobody w doborze rozdzielczości bitowej w zalezności od

zastosowań. Przeważnie jednak w zastosowaniach amatorskich rodzielczość przekraczająca 12 bitów rzadko bywa potrzebna, a precyzyjne przetworniki 16- i 24-bitowe są dosyć drogie.
Wszystkie pomiary, bez względu na sposób ich wykonania, zasadę i konstrukcje układu są obarczone pewnym błędem. Przyczyny niedokładności pomiarów i ich wielkość są zależne od szeregu czynników. Na dokładność pomiaru wpływają takie czynniki wewnętrzne jak odchyłki charakterystyki od idealnej (przykładowo jej nieliniowości), starzenie się czujnika i pozostałych elementów układu i związane z tym zmiany ich parametrów, czynniki zewnętrzne takie jak zmiany temperatury otoczenia, zakłócenia elektryczne wpływające na wartość otrzymej wielkości elektrycznej oraz oczywiście błędy wynikające z samej zasady pomiaru i wpływu układu pomiarowego na badany obiekt i mierzoną wielkość. Przykładowo układ termometru elektronicznego może dostarczać ciepła do badanego obiektu lub też je od niego pobierać zmieniając w ten sposób jego temperaturę, a włączony do obwodu amperomierz powoduje zmianę płynącego w nim prądu. W momentach zmiany wartości mierzonej dodatkowo mogą występować też błędy dynamiczne jeżeli układ pomiarowy nie nadąża za zmianą wielkości mierzonej (ma zbyt dużą stałą czasu).
W rozwiązaniach cyfrowych dodatkowo do nich dochodzą błędy wynikające z kwantyzacji przy przetwarzaniu danych z postaci analogowej na cyfrową (błąd +/-1 na ostatniej pozycji, nieliniowości charakterystyki przetwornika, ograniczona rozdzielczość przetwornika) i niedokładności powstające w wyniku przeliczeń np. w skutek ograniczonego zakresu liczbowego procesora. Ogólnie rzecz biorąc błędy pomiaru można podzielić w zależności od przyczyn na dwie kategorie: błędów systematycznych i błędów przypadkowych.
Niedokładności (błędów) pomiarowych nie można uniknąć całkowicie, można jedynie starać się je zminimalizować, i to tylko w stopniu zależnym od potrzeby i przewidzianych nakładów pracy i finansowych.
Powszechnie stosowaną metodą minimalizacji wpływu błędów przypadkowych i w pewnym stopniu także dynamicznych jest wykonywanie serii pomiarów i obliczanie wartości średniej. Stanowi to swego rodzaju rachunkowy filtr dolnoprzepustowy.

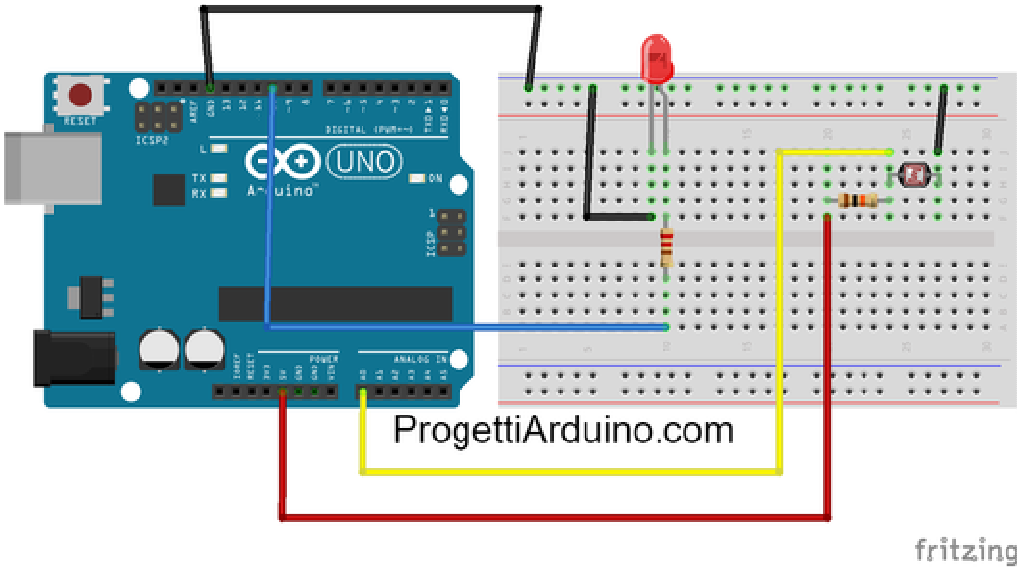


Rys. 1.1.2. Przykładowe czujniki pracujące w układzie dzielnika oporowego, od lewej do prawej: fotoopornik, termistor, fofodioda, fototranzystor, potencjometr obracany przez wiatrowskaz lub inne urządzenie mechaniczne

Zastosowane w dalej opisanych rozwiązaniach mikrokomputery w rodzaju „Arduino”, „Maliny” i podobne służą nie tylko do przeliczenia otrzymanych danych na rzeczywiste jednostki, ale także do tworzenia z nich komunikatów w wybranych formatach, nadawania ich i sterowania radiostacjami.
W dalszym ciągu rozdziału przedstawiono przykłady popularnych i nieskomplikowanych rozwiązań pomiarów niektórych wielkości nieelektrycznych.

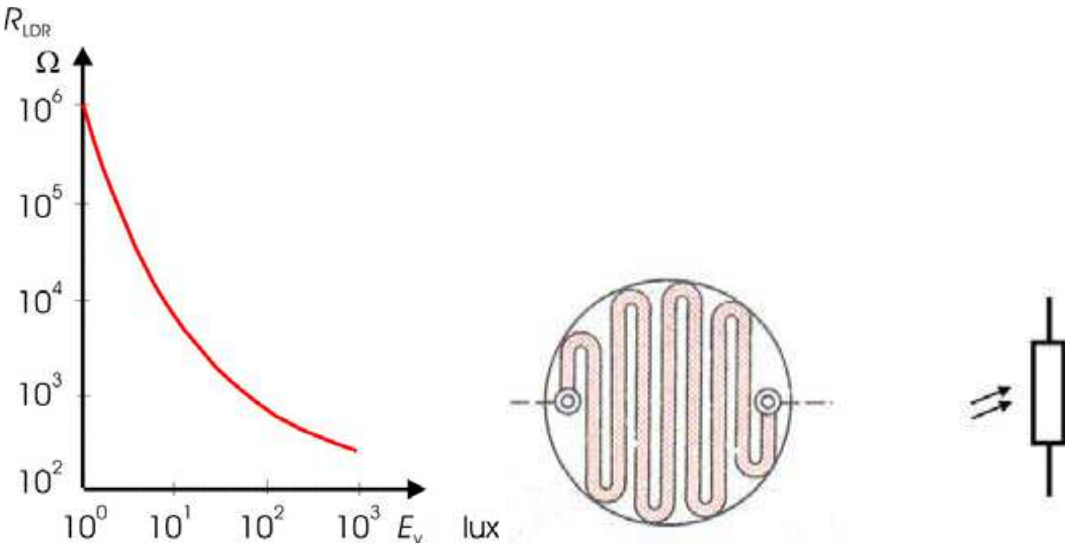
Pomiar oświetlenia

Do pomiaru oświetlenia używane są często fotooporniki włączone w układ dzielnika oporowego, którego drugim elementem jest opornik stały. Przy zasilaniu dzielnika stałym i dobrze stabilizowanym napięciem pomiar oświetlenia elementu światłoczułego sprowadza się do pomiaru napięcia na wyjściu dzielnika. Przedstawiony na ilustracji układ ilustrujący zasadę pomiaru został skonstruowany na uniwersalnej płytce eksperymentalnej. Jasność czerwonej diody świecącej jest proporcjonalna do zmierzonej jasności otoczenia. W szereg z diodą włączony jest opornik 220 Ω, a w szereg z fotoopornikiem – 100 kΩ. Układ można zastosować m.in. do pomiaru liczby godzin słonecznych. Pomiar ten polega na zliczaniu odcinków 1/10 godziny, w których oświetlenie słoneczne przekracza 120 W/m^2.
Analogicznie przy pomiarze temperatury jednym z elementów dzielnika jest termistor. Przykład poniższy ilustruje ogólnie sposób korzystania z czujników pracujących na zasadzie dzielnika napięcia. Oprócz fotooporników stosowane są też fotodiody i fototranzystory. Charakteryzują się one znacznie krótszym czasem reakcji, a fototranzystory zapewniają dodakowo wzmocnienie sygnału.

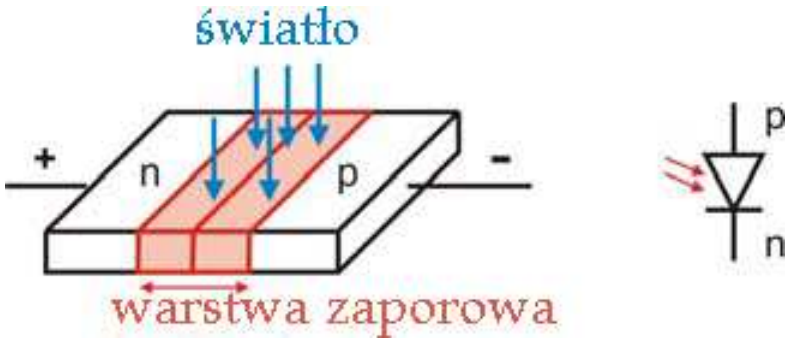


Rys. 1.2.1. Przykład pomiaru jasności oświetlenia za pomocą „Arduino” przy wykorzystaniu fotoopornika (źródło: *www.progettiarduino.com*)

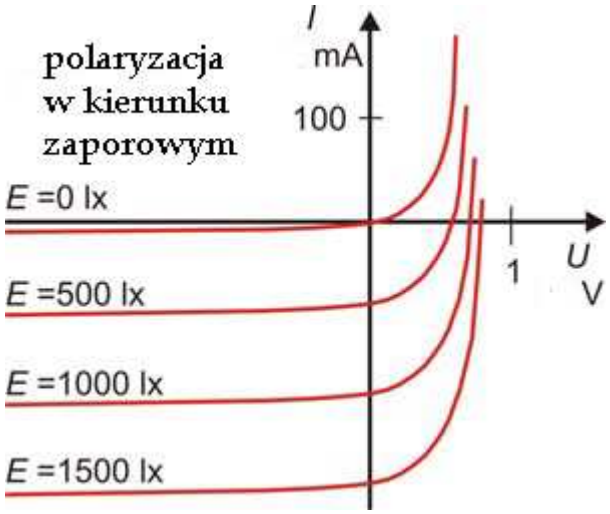
```
// Arduino dioda świecąca i fotoopornik, jasność diody zależna od oświetlenia
// Dalsze informacje: www.ProgettiArduino.com

int luminosita;  // Oporność fotoopornika
int ledPin = 10;  // Wyprowadzenie diody świecącej

void setup() {
}

void loop() {
  luminosita = analogRead(A0);  // Odczyt jasności
  luminosita = luminosita/4;  // Dopasowanie skali
  analogWrite(ledPin,luminosita);  // Wydanie wartości na diodę
  delay(10);  // Opóźnienie 10 ms
}
```



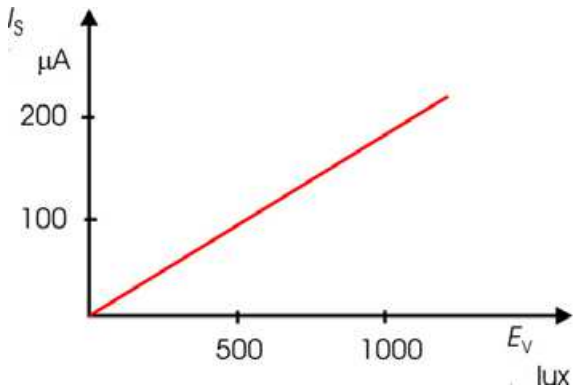
Rys. 1.2.2. Przykładowa charakterystyka fotoopornika, jego wygląd i symbol



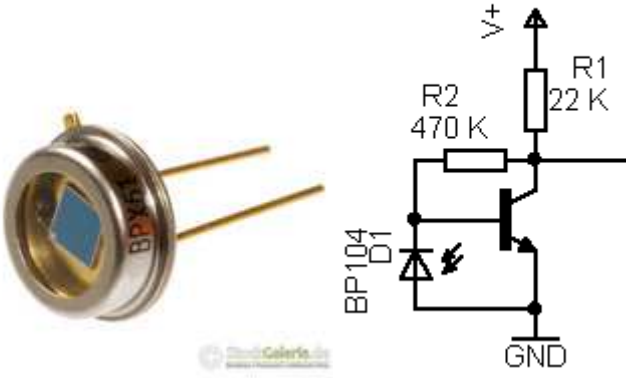
Rys. 1.2.3. Fotodioda – zasada pracy i symbol. Padające na złącze fotony powodują powstawanie par elektron-dziura



Rys. 1.2.4. Przykładowe charakterystyki fotodiody spolaryzowanej w kierunku zaporowym.

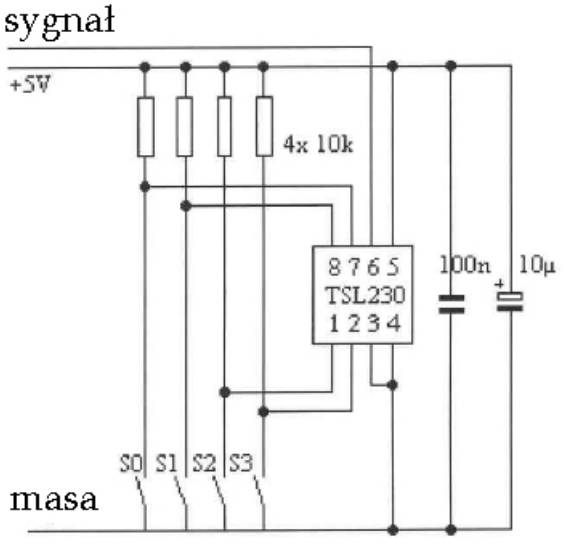


Rys. 1.2.5. Przykładowa zależność prądu w kierunku zaporowym od oświetlenia diody



Ilustr. 1.2.6. Fotodioda i prosty układ wzmacniacza dla niej

Przykładem elementów przetwarzających mierzoną wielkość na częstotliwość jest przetwornik światło-częstotliwość TSL235 (i jego odpowiedniki lub nowsze rozwiązania).



Rys. 1.2.7. Układ pomiarowy dla TSL230

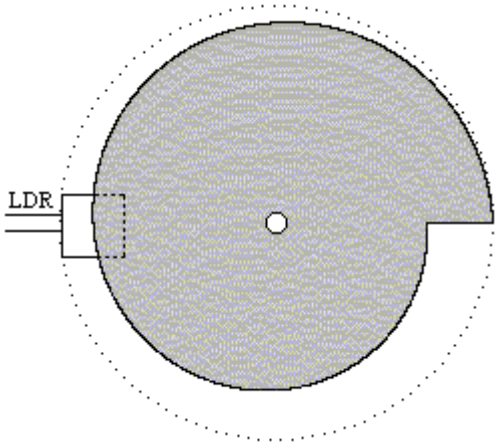
Dokładność pomiaru dla TSL230 wynosi 20%, TSL230A – 10% iTSL230B – 5%.

Tabela 1.2.1
Położenia przełączników S0 – S3 dla TSL230

S0	S1	Czułość	S2	S3	Częstotliwość
0	0	Wyłączony	0	0	0 – 1 MHz
0	1	x 1	0	1	0 – 500 kHz
1	0	x 10	1	0	0 – 100 kHz
1	1	x 100	1	1	0 – 10 kHz

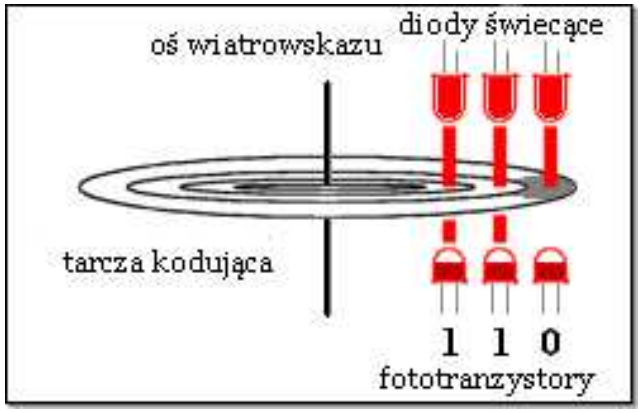
TSL235 nie posiada możliwości przełączania zakresów i dostarcza sygnału o częstotliwości 0 – 500 kHz.
Do pomiaru czasu i jasności nasłonecznienia można zastosować także małe ogniwo słoneczne.

Wskazania kierunku wiatru

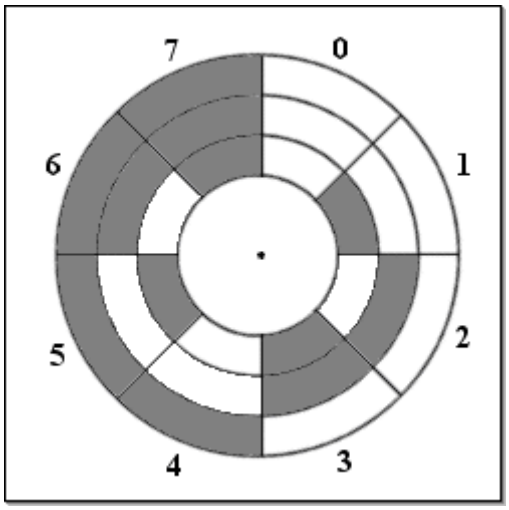


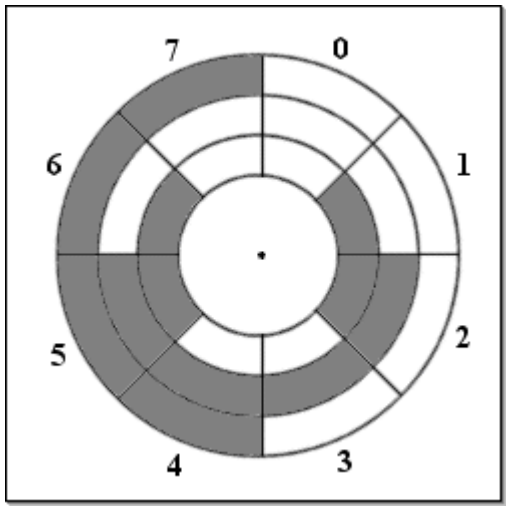
Rys. 1.3.1. Przysłona do pomiaru kierunku wiatru metodą optyczną.

Widoczna na ilustracji powyżej przysłona zasłania fotooporник w mniejszym lub większym stopniu co powoduje zmianę intensywności jego oświetlenia, a co za tym idzie jego oporności. Do oświetlenia fotooporника może służyć umieszczona po drugiej stronie tarczy dioda elektroluminescencyjna DEL (ang. LED). Tarcza jest umocowana na wspólnej osi z wiatrowskazem i powinna być umieszczona w światłoszczelnej obudowie.
Kształt tarczy jest opisany następującym wzorem (dla współrzędnych biegunowych):
$r(\alpha) = 1 + \alpha/c$ gdzie α jest kątem wyrażonym w radianach, r – promieniem, a c – parametrem opisującym odchyłkę tarczy od okręgu.
Maksymalny błąd pomiaru występuje w miejscu skoku średnicy, dlatego też powinien on odpowiadać możliwie rzadko występującemu kierunkowi wiatru np. północnemu lub wschodniemu.



Rys. 1.3.2. Zasada pomiaru przy użyciu tarczy kodującej





Rys. 1.3.3. Tarcza z kodem dwójkowym Rys. 1.3.4. Tarcza z kodem Greya

Do pomiaru kierunku wiatru można użyć także tarczy kodującej. Zaletą tego rozwiązania jest uzyskiwanie od razu informacji cyfrowej, którą można bezpośrednio wykorzystać w programach komputerowych bez konieczności dokonywania dodatkowych przeliczeń. Zasadę pomiaru ilustruje rysunek 1.3.2. Tarcza kodująca składająca się z odpowiednio rozmieszczonych segmentów przezroczystych i nieprzezroczystych jest umieszczona na osi wiatrowskazu. Pomiar dokonywany optycznie nie wymaga żadnych zużywających się kontaków ani innych elementów.
Kierunki mogą być zakodowane dwójkowo (rys. 1.3.3) lub przy użyciu kodu Greya (rys. 1.3.4). O ile w pierwszym przypadku skoki wartości między segmentami kierunkowymi mogą wynosić od jednego do trzech bitów, o tyle w drugim różnice wynoszą zawsze tylko jeden bit, co zmniejsza ryzyko błędnego odczytu (ułatwia jego rozpoznanie).

Pomiar szybkości wiatru

Rys. 1.4.1. Wiatromierz

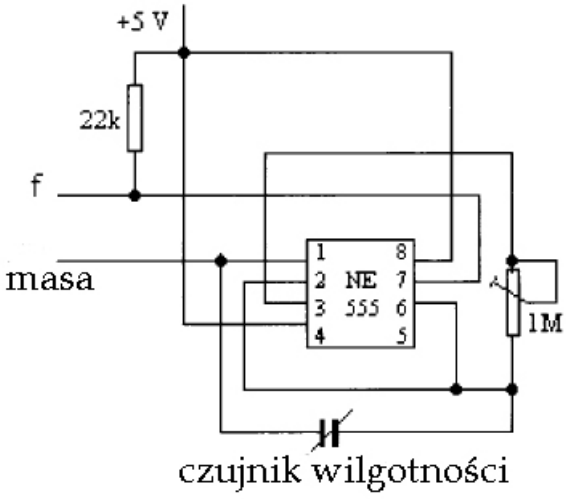
Do pomiaru szybkości wiatru wystarczy użyć wiatraczka złożonego z trzech elementów. Można go umocować na osi silniczka elektrycznego pracującego w tej konfiguracji jako generator dostarczający napięcia zależnego od szybkości obrotów.
Inną możliwością jest umieszczenia na osi tarczy posiadającej naprzemian segmenty przepuszczające i załaniajace światło i pomiar częstotliwości impulsów przy użyciu fotoelementu (fotodiody itp.) i mikrokomputera. W przypadku pomalowania nieprzezroczystej tarczy na przemian na kolory biały i czarny wykorzystuje się światło od niej odbite.

Rys. 1.4.2. Tarcza przerywająca strumień światła

Pomiar wilgotności

Fot. 1.5.1. Czujnik wilgotności powietrza

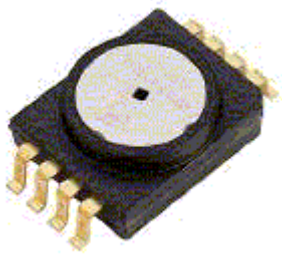
Do często stosowanych rozwiązań należy pomiar wilgotności na zasadzie pojemnościowej Czujnik zawiara dwie elektrody odizolowane od siebie za pomocą folii zmieniajacej stałą dielektryczną w zależności od stanu zawilgocenia. Powstaje w ten sposób kondensator o pojemności zależnej od wilgotności otoczenia (przykładowo w zakresie 100 – 140 pF). Kondensator ten można włączyć do obwodu drgań gerneratora (np. na LM555) i dokonywać pomiaru jego częstotliwości.



Rys. 1.5.2. Generator o częstotliwości zależnej od wilgotności

Innymi popularnymi typami czujników wilgotności (lub temperatury i wilgotności) są SEN-08251, SEN-09569, SEN-10239, SEN-11295, KFS140, KFS330, HYT221 i HYT271.

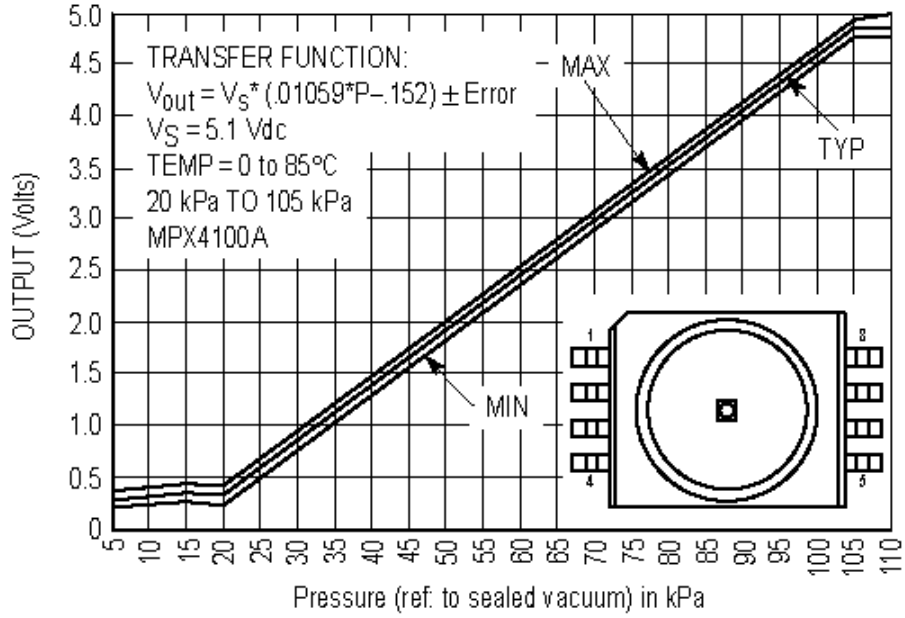
Pomiar ciśnienia atmosferycznego



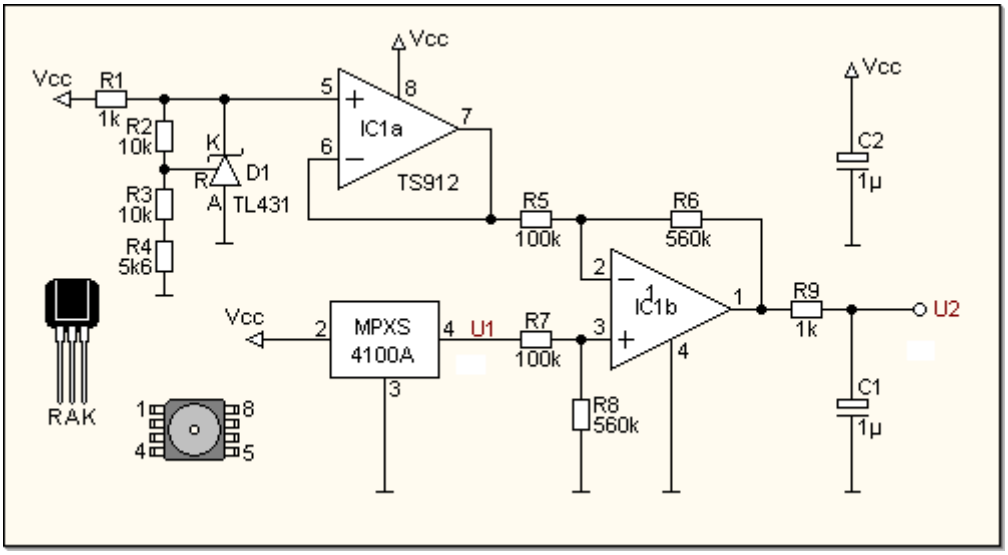
Fot. 1.6.1. Piezoelektryczny barometr MPXS54100

W czujniku pracującym na zasadzie piezoelektrycznej powietrze dochodzące do wnętrza przez otworek oddziaływuje poprzez cienką folię na materiał piezoelektryczny dostarczający na zaciskach napięcia

proporcjonalnego do ciśnienia. Po drugiej stronie folii znajduje się komora wypełniona powietrzem pod ciśnieniem wzorcowym.
Charakterystykę barometru MPX4100A przedstawionio na ilustracji 1.6.2. Pracuje on w zakresie 200 – 1050 hPa (20 – 105 kPa). Dokładność pomiaru wynosi +/- 1,8% w zakresie temperatur 0 – 85°C.
Napięcie wyjściowe czujnika można doprowadzić do wejścia analogowego „Arduino” lub innego mikrokomputera bezpośrednio albo też odjąć od niego pewną stałą wartość dla zwiększenia dokładności przetwarzania na postać cyfrową.
Dla ciśnień w zakresie 920 – 1050 hPa czujnik dostarcza napięcia (typ.) od 4,1124 do 4,80075 V. Dla lepszego wykorzystania rozdzielczości przetwornika analogowo-cyfrowago można odjąć od niego napięcie 4,096 V np. w układzie przedstawionym na ilustracji 1.6.3. Na jego wyjściu otrzymuje się więc napięcie 0,09184 – 3,9466 V. Napięcie odniesienia 4,096 V otrzymywane jest w układzie stabilizatora TL431.



Rys. 1.6.2. Charakterystyka barometru MPX4100A



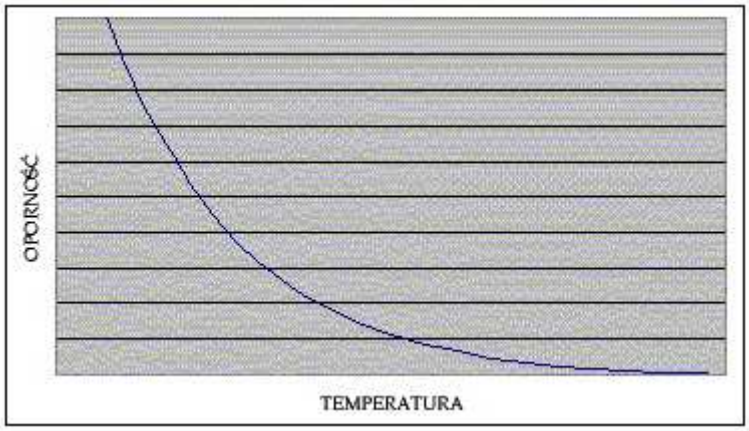
Rys. 1.6.3. Układ pomiarowy z MPXS4100A

Przy podanych na schemacie stosunkach R6/R5 i R7/R8 wzmocnienie IC1b wynosi 5,6.
Do innych często stosowanych czujników barometrycznych należą BMP085 firmy „Bosch”, oraz MS5607, MS5611 i MS5803.

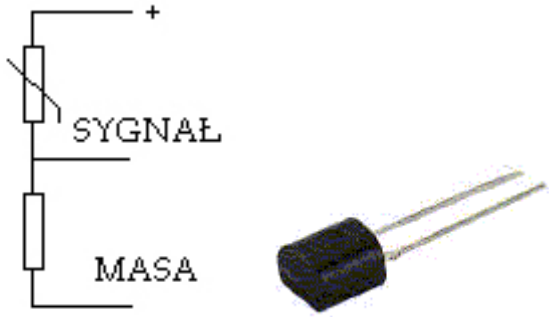
Fot. 1.6.4. Czujniki barometryczne

Pomiary temperatury

Najprostszym sposobem pomiaru temperatury jest pomiar przy użyciu termistora czyli opornika o oporności zależnej od temperatury. Przykładową charakterystykę termistora przedstawiono na ilustracji 1.7.1. Zależność ta jest nieliniowa przez co dla otrzymania temperatury konieczne jest wykonanie przedstawionych w dalszych rozdziałach obliczeń. Oprócz termistorów o charakterystyce opadającej w funkcji temperatury produkowane są też termistory o wzrastającej oporności. Jeden i drugi rodzaj znajdują szerokie zastosowanie w elektronice.

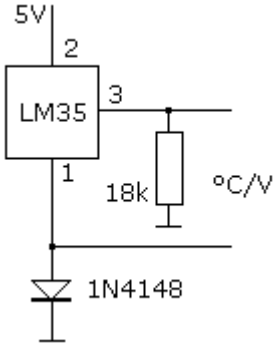


Rys. 1.7.1. Zależność oporności termistora od temperatury

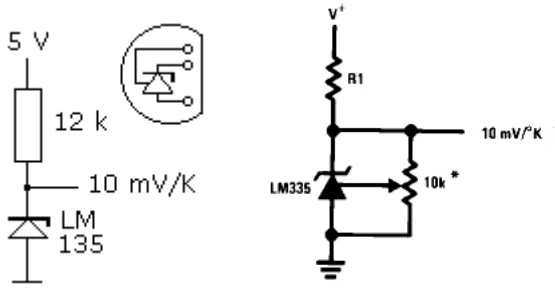


Rys. 1.7.2. Układ pomiarowy – dzielnik napięcia i wygląd termistora z serii KTY81–84

Czujniki LM35 są fabrycznie skalibrowane i dostarczają napięcia 10 mV na stopień Celsjusza co ułatwia przetworzenie danych i odczyt panującej temperatury. Dla temperatur ujemnych konieczne byłoby jednak zasilanie układu napięciem ujemnym i przetwarzanie dalej tego napięcia. Na schemacie 1.7.3. przedstawiono elegancki sposób rozwiązania tego problemu.

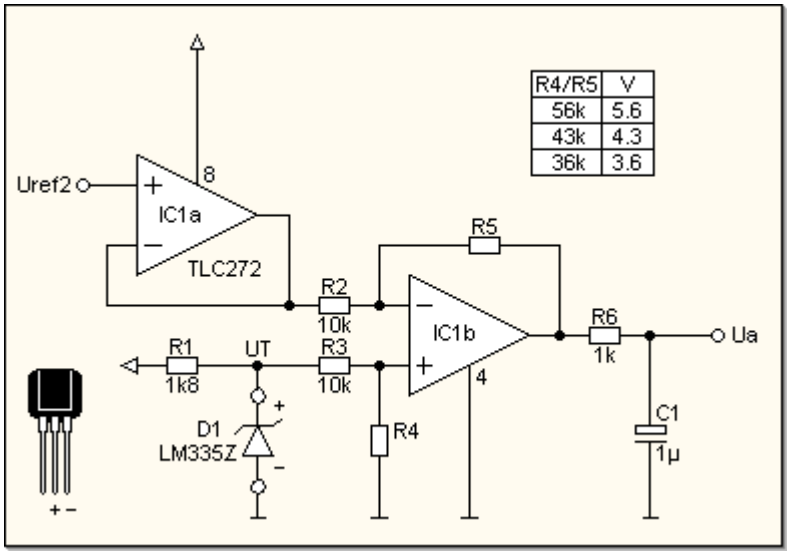


Rys. 1.7.3. Układ pomiarowy dla ujemnych temperatur na LM35



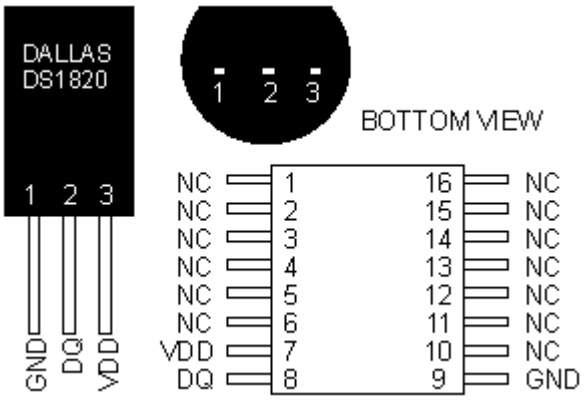
Rys. 1.7.4. Schemat podłączenia, wyprowadzenia LM135/LM235/LM335 i układ z kompensacją

Czujniki LM135, LM235 i LM335 zawierają układ odpowiadający skompensowanej diodzie Zenera i bez dodatkowej kompensacji zapewniaja dokładność +/- 1ºC. Zależność napięcia wyjściowego od temperatury jest identyczna jak dla LM35 czyli 10 mV/ ºC. LM135 pokrywa zakres -55 – +150 ºC, LM235 – zakres -40 – +125 ºC, a LM335 -40 – +100 ºC. Dla uzyskania większej dokłaności należy równolegle do czujnika podłączyć potencjometr 10 kΩ i jego suwak połączyć z wejściem kompensacyjnym.



Rys. 1.7.5. Układ pomiarowy z LM335

Układ z rys. 1.7.5. jest przewidziany do pomiarów w zakresie od -20 ºC. Temperaturze tej odpowiada napięcie 2,5315 V, dlatego też dla lepszego wykorzystania rozdzielczości przetwornika analogowo-cyfrowego w mikrokomputerze odejmowane jest od niego napięcie 2,5 V. Korzystnie jast wybrać takie wzmocnienie IC1b (stosunek R4/R5), aby dla górnej granicy pożądanego zakresu pomiarowego napięcie wyjściowe wynosiło około 4 V.



Rys. 1.7.6. Wyprowadzenia czujników DS1820 i podobnych

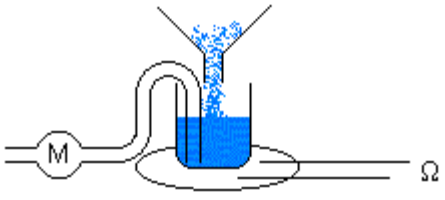
Czujniki DS1820 i pokrewne pracują na zupełnie innej zasadzie. Dla zmierzenia temperatury zliczane są przejścia przez zero sygnału oscylatora o niskim współczynniku temperaturowym (czyli częstotliwość jego drgań). Dane przekazywane są przez złącze jednoprzewodowe (*One Wire*, *1-Wire*) w postaci 9-bitowych bloków. Pierwszy z bitów oznacza znak – 0 dla wartości dodatnich i 1 dla ujemnych. Pozostałe 8 bitów zawiera zmierzoną wartość w krokach po 0,5 ºC. Wartości ujemne są przedstawione w postaci uzupełnienia dwójkowego. Temperaturze 0,5 ºC odpowiada więc wartość 1, temperaturze 25 ºC – wartość 50 (32 szesnastkowo), a temperaturze -0,5 ºC – 255 (FF szesnastkowo). Czujniki te są przewidziane w pierwszym rzędzie do podłączenia do mikrokomputerów. Każdy z nich posiada 48-bitowy numer seryjny umożliwiający jednoznaczną identyfikację w układach zawierających większą liczbę czujników. Nazwa złącze jednoprzewodowe jest oczywiście nazwą umowną lub fabryczną ponieważ nawiązuje do użycia tylko jednego przewodu sygnałowego i nie uwzględnia niezbędnego połączenia masy.
Do pomiaru temperatury służą także termopary (elementy złożone z dwóch różnych połączonych ze sobą metali) dające napięcie zależne od temperatury, a dokładnie od różnicy temperatur między punktem pomiarowym, a punktem odniesienia majacym znaną temperaturę
$U = k (T – T_0)$, gdzie k jest współczynnikiem zależnym od obu materiałow, a T_0 temperaturą odniesienia.
W zależności od szybkości wiatru i powodowanego przez to dodatkowego chłodzenia ludzkiej skóry temperatura odczuwalna jest w takiej sytuacji niższa od zmierzonej w sposób standardowy tzn. w osłoniętym miejscu, w cieniu i na wysokości 2 m nad powierzchnią ziemi. Do obliczania odczuwalnej temperatury używany jest m.in. następujący wzór NOAA:

$$T_C = 55{,}63 + 1{,}1187 * T - 23{,}82 * V^{0{,}16} + 0{,}8303 * T * V^{0{,}16}$$

Gdzie T_C jest temperaturą odczuwalną, T – rzeczywiście zmierzoną, a V – szybkością wiatru.

Pomiary ilości opadów

Woda z opadów zbierana jest w naczyniu przez ustalony czas,a następnie odczytywana jest jej objętość, a właściwie wysokość słupa wody w mm. Dla zminimalizowania błędów pomiaru konieczne jest zbieranie opadów z możliwie większej powierzchni korzystając z lejka. W przypadku pomiaru elektronicznego możliwe jest podłożenie pod pojemnik czujnika nacisku np. przedstawionego na ilustracji czujnika foliowego. Oporność między jego wyprowadzeniami zmienia się zależnie od nacisku.



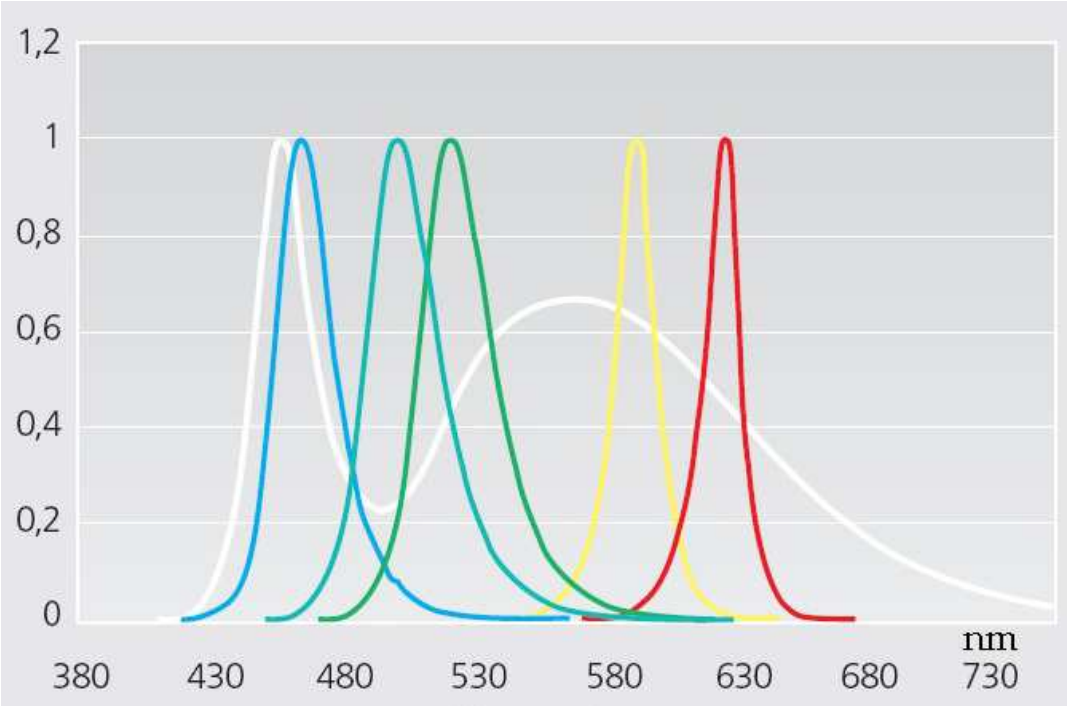
Rys. 1.8.1. Zasada pomiaru

Rys. 1.8.2. Foliowy czujnik nacisku

Inną szeroko rozpowszechnioną metodą pomiaru jest metoda z ruchomą łyżeczką. Po zapełnieniu się pojemnika-łyżeczki opada ona opróżniając się i generuje impuls elektryczny. Całkowitą objętość opadów otrzymuje się przez zliczanie impulsów w wybranym czasie.

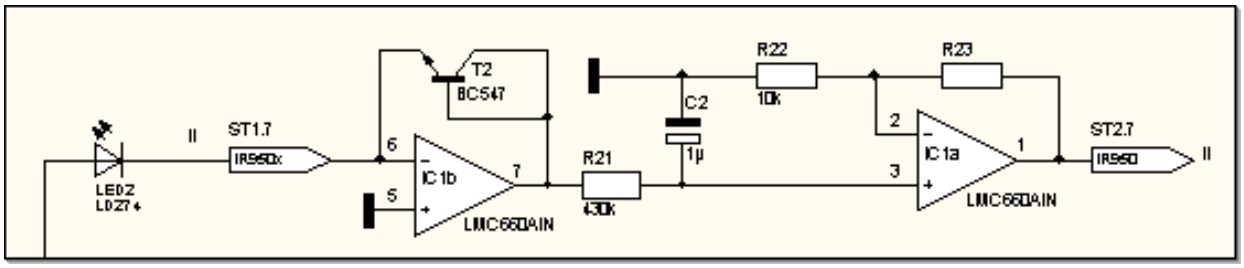
Spektrometr

Do pomiaru koloru nieba i chmur w dzień w różnych warunkach atmosferycznych i w nocy można użyć kolorowych diod elektroluminescencyjnych. Ich charakterystyki czułości widmowej są zależne od koloru świecenia. Przykładowe charakterystyki promieniowania różnokolorowych diod przedstawiono na ilustracji 1.9.1. Rzeczywiste przebiegi charakterystyk czułości jako fotodiody zależą od typu diody i rodzaju (koloru) obudowy.



Rys. 1.9.1. Orientacyjne charakterystyki widmowe kolorowych diod elektroluminescencyjnych i diody białej

Na schemacie 1.9.2 przedstwiony jest tor pomiarowy o charakterystyce logarytmicznej. Całkowity układ składa się z kilku takich torów – po jednym dla każdej z użytych diod. Napięcia wyjściowe torów należy doprowadzić do wejść analogowych mikrokomputera. Elementy R21 i C2 tworzą filtr dolno-przepustowy o częstotliwości granicznej 0,4 Hz. Ze względu na różnice wartości napięć dostarczanych przez diody konieczne jest dopasowanie wzmocnienia wzmacniacza IC1a przez dobór opornika R23. Wzmacniacz operacyjny jest zasilany napięciami +/-.

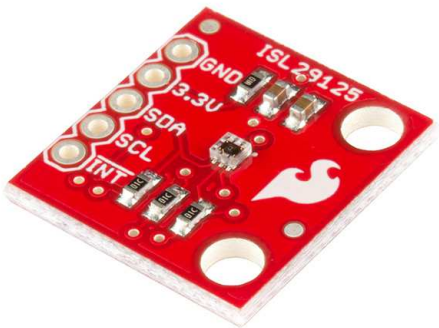


Rys. 1.9.2. Schemat pojedyńczego toru pomiarowego. Anoda diody jest połączona z masą

Na rys. 1.9.3. przedstawiony jest pełny schemat 7-kanałowego spektrometru. Składa się on z następujących kanałów (od góry do dołu):

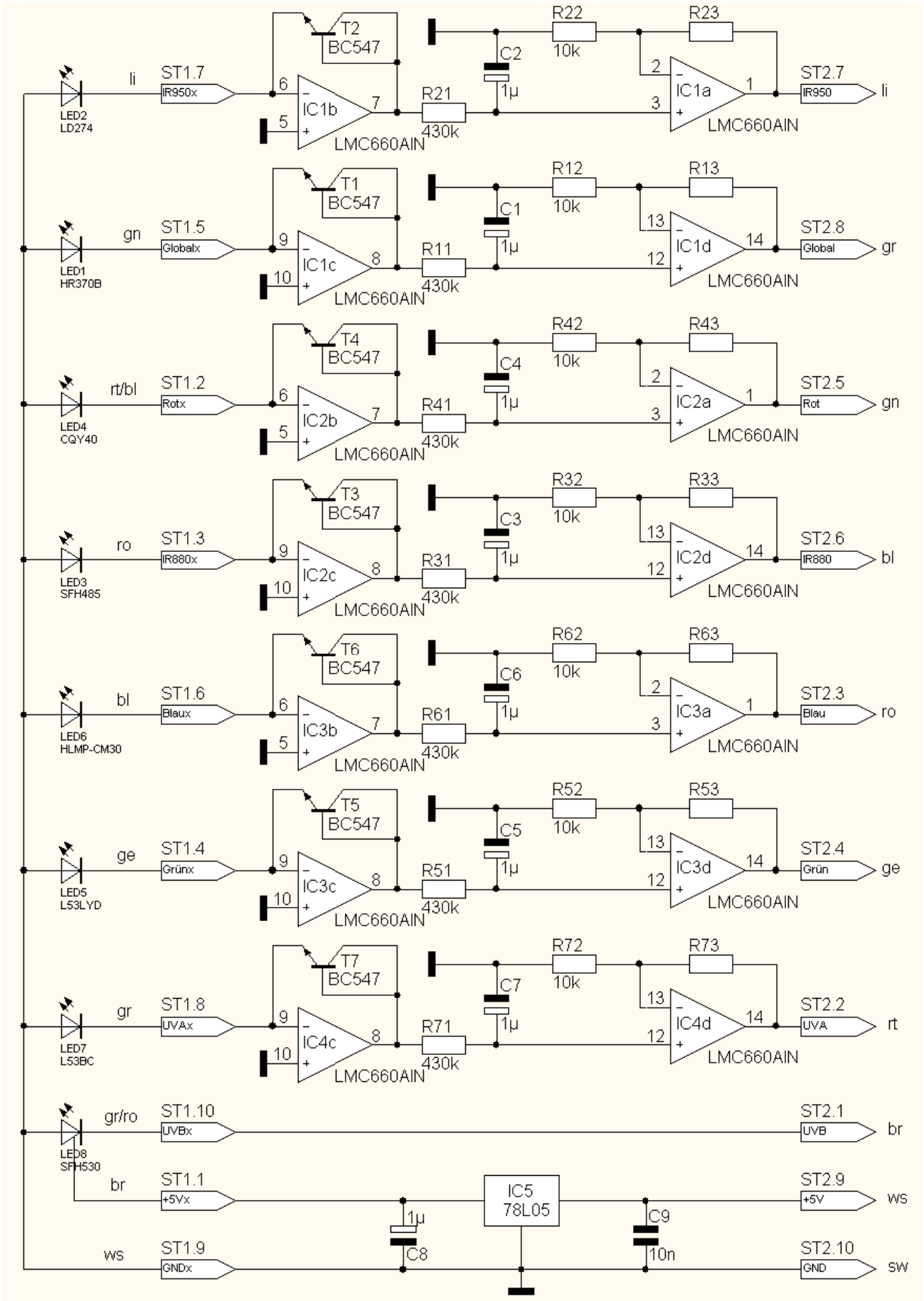
- Kanał podczerwieni 950 nm,
- Kanał jasności całkowitej,
- Kanał czerwieni,
- Kanał podczerwieni 880 nm,
- Kanał błękitu,
- Kanał zieleni,
- Kanał promieniowania ultrafioletowego A,
- Kanał promieniowania ultrafioletowego B.

W zależności od potrzeb i możliwości można oczywiście zrezygnować z niektórych z nich, ograniczając się w najprostszym przypadku tylko do trzech podstawowych: czerwieni, błękitu i zieleni. Również przy zastosowaniu jedynie diod czerwonej i niebieskiej można przeprowadzić interesujące doświadczenia. Oprócz analizy światła dziennego, koloru nieba i chmur interesująca może okazać się też analiza różnych płynów na podstawie kolorów przy oświetleniu różnymi źródłami światła lub analiza porównawcza – przez porównywanie widma z widmami światła przepuszczanego przez znane płyny przyjęte jako wzorcowe.
W handlu dostępne są fotodiody o różnych charakterystykach czułości (np. charakterystyce dla światła czerwonego i niebieskiego albo czerweonego, zielonego i niebieskiego) umieszczone we wspólnej obudowie. Nadają się one bardzo dobrze do takiej analizy widmowej w badaniach środowiska, w analityce, w medycynie, w przemyśle itd.
Jednym z takich czujników jest ISL29125. Komunikacja z nim odbywa się za pośrednictwem złącza I2C. Jego charakterystyka czułości dla trzech kolorów podstawowych jest dostosowana do czułości oka ludzkiego. Wykazuje on wprawdzie w pewnym stopniu także czułość na promieniowanie ultrafioletowe A, ale w zastosowaniach, w których jest to niepożądane wystarczy przysłonięcie go zwykłą szybką szklaną. Obudowa ISL29125 ma wymiary 1,65 x 1,65 mm. Na szczęście dostępne są też gotowe płytki drukowane ułatwiające połączenie czujnika z „Arduino” lub innym mikrokomputerem.



Fot. 1.9.4. Płytka czujnika ISL29125

Specjalnie dla „Arduino” został opracowany moduł zawierający czujnik RGB typu TCS34725. Moduł jest przystosowany do zasilania napięciem 3,3 V. Biblioteka do komunikacji „Arduino” z nim jest dostępna na witrynie *Github*.



Rys. 1.9.3. Spektrometr 7-kanałowy

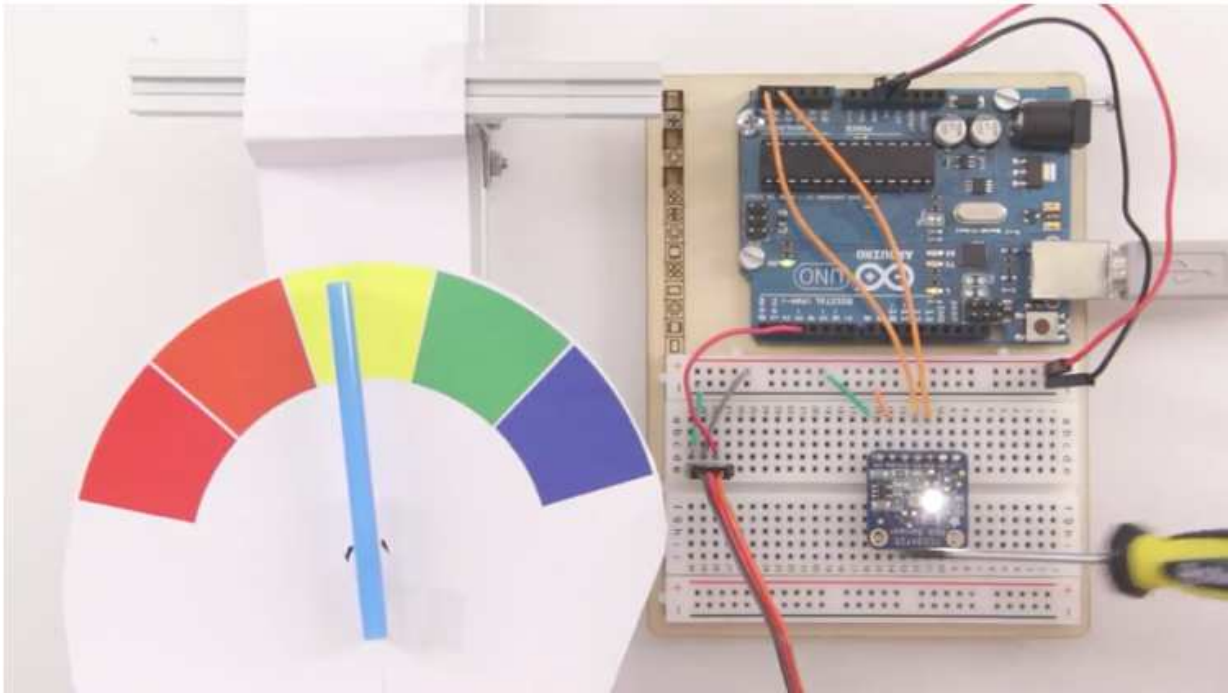
Przykładowy program do odczytu TCS34725 przez „Arduino” (*www.makerblog.at*):

```
// Włączenie bibliotek
#include "Servo.h"
#include "Wire.h"
#include "Adafruit_TCS34725.h"

// Pozycje silnika dla poszczególnych kolorów, wymagają dopasowania
const int redPos = 160;
const int orangePos = 130;
const int yellowPos = 100;
const int greenPos = 70;
const int bluePos = 30;
const int nonePos = 0; // Obiekt nie rozpoznany

// Założenie obiektu serwo
Servo myservo;

// Inicjalizacja obiektu czujnika kolorów
// Parametry pod: https://learn.adafruit.com/adafruit-color-sensors/program-it
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_1X);

// funkcja setup() wywoływana jednorazowo przy starcie programu na Arduino
void setup() {

 // Złącze szeregowe do wyświetlania danych w oknie monitora
 Serial.begin(9600);
 Serial.println("Makerblog.at - MuMs Color Sensor");

 // Sprawdzenie funkcjonowania czujnika kolorów
 if (tcs.begin()) {
 // Wszystko w porządku
 Serial.println("Sensor gefunden");
 } else {
 // Nie znaleziono czujnika, zatrzymanie programu w pętli
 Serial.println("TCS34725 nicht gefunden ... Ablauf gestoppt!");
 while (1); // Stop!
 }

 // Silniczek serwo podłączony do wyprowadzenia PWM 3
 myservo.attach(3);
 // Sprowadzenie silniczka do położenia wyjsciowego
 myservo.write(0);
 delay(1000);

}

// funkcja loop() wykonywana ciągle
void loop() {

 // Czujnik dostarcza wartości dla kolorów R, G, B i jasności
 uint16_t clearcol, red, green, blue;
 float average, r, g, b;
 delay(100); // Pomiar koloru trwa ok. 50 ms
```

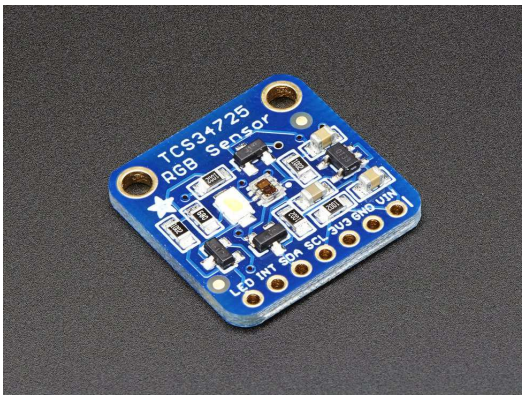
```
tcs.getRawData(&red, &green, &blue, &clearcol);

// Próba rozpoznania koloru obiektu.
// Skala kolorów: czerwony, zielony, niebieski, pomarańczowy i żółty

// Obliczenie wartości średniej dla RGB
average = (red+green+blue)/3;
// Wartości kolorów normalizowane przez podział przez średnią,
// Wyniki w okolicach jedności
r = red/average;
g = green/average;
b = blue/average;

// Poziomy jasności i kolorów wydane do monitora dla kontroli
// r,g i b w przybliżeniu pomiedzy 0,5 i 1,5
// Dla obiektór czerwonych r powinno leżeć wyraźnie powyżej 1,0
// a g i b pomiedzy 0,5 i 1,0 itd.
Serial.print("\tClear:"); Serial.print(clearcol);
Serial.print("\tRed:"); Serial.print(r);
Serial.print("\tGreen:"); Serial.print(g);
Serial.print("\tBlue:"); Serial.print(b);

// Próba określenia koloru na podstawie wartości r, g i b
// Zalecane jest rozpoczecie rozpoznawania od kolorów czerwonego, zielonego i niebieskiego
// i odpowiednie dopasowanie granic w programie
if ((r > 1.4) && (g < 0.9) && (b < 0.9)) {
Serial.print("\tROT");
myservo.write(redPos);
}
else if ((r < 0.95) && (g > 1.4) && (b < 0.9)) {
Serial.print("\tGRUEN");
myservo.write(greenPos);
}
else if ((r < 0.8) && (g < 1.2) && (b > 1.2)) {
Serial.print("\tBLAU");
myservo.write(bluePos);
}
// Żółć i pomarańcz są trochę trudniejsze do rozpoznania, ale  w próbach sprawdziły się
// poniższe wartości
else if ((r > 1.15) && (g > 1.15) && (b < 0.7)) {
Serial.print("\tGELB");
myservo.write(yellowPos);
}
else if ((r > 1.4) && (g < 1.0) && (b < 0.7)) {
Serial.print("\tORANGE");
myservo.write(orangePos);
}
// dla kolorów nie rozpoznanych
else {
Serial.print("\tNICHT ERKANNT");
// myservo.write(nonePos);
}

// nowa linia
Serial.println("");
```

```
 // Odstęp czasu dla prób
 delay(100);

}
```

W przykładzie modelarski silniczek „serwo” obraca wskazówkę pokazującą rozpoznany kolor na kolorowej 5-stopniowej skali. W oknie monitora szeregowego wyświetlane są zmierzone wielkości odpowiadające trzem podstawowym kolorom.



Fot. 1.9.5. Demonstracyjny układ rozpoznawania kolorów. Obok fotodiody widoczna jest dioda oświetlająca badany przedmiot



Fot. 1.9.6. Płytka TCS34725 (źródło: *www.adafruit.com*)

Gotowe czujniki pomiarowe

Czujniki „Iduino”

Seria fabrycznych czujników „Iduino” jest przystosowana do współpracy z jednopłytkowymi mikrokomputerami „Arduino”. Ich zaletą jest stosunkowo niska cena, różnorodność i łatwość wykorzystania we własnych konstrukcjach eksperymentalnych. Rozwiązania te mogą także stać się inspiracją dla własnych konstrukcji czujników pomiarowych nie tylko dla „Arduino” ale również i dla „Maliny” albo podobnych mikrokomputerów. Dokładność pomiarów jest w zupełności wystarczająca nie tylko do zastosowań szkolnych (dydaktycznych), do czego są one przewidziane w pierwszym rzędzie, ale również i do innych celów amatorskich, w tym także krótkofalarskich transmisji danych meteorologicznych albo innych w systemach APRS, DPRS, „LoRa” i temu podobnych.
Oferta układów i manipulatorów „Iduino” jest znacznie szersza od przedstawionej poniżej. Autor ograniczył się tutaj jedynie do czujników pomiarowych pomijając klawisze, manipulatory, mikrofony, układy sygnalizacyjne, wykonawcze i inne.
Załączone przykładowe programy lub ich fragmenty pochodzą z dokumentacji urządzeń. W oparciu o nie można stosunkowo łatwo opracować własne podprogramy wplecione w większą całość. W niektórych przypadkach konieczne jest zainstalowanie dodatkowych bibliotek dla „Arduino” w sposób podany w tomach 20 i 21 „Biblioteki polskiego krótkofalowca” lub w innych instrukcjach do „Arduino”. Część z nich dostępna jest w internecie pod adresem [1], a pozostałych przypadkach można pod nim przeważnie trafić na właściwy trop. O nazwach potrzebnych bibliotek informują najczęściej nazwy włączanych (za pomocą poleceń #include) do programów plików nagłówkowych. O braku niezbędnej biblioteki informuje też kompilator wskazując w meldunkach błędów na związany z nią plik nagłówkowy.

Wykrywacz płomieni

Rys. 2.1.1. Wykrywacz płomieni

Wykrywacz reaguje na promieniowanie podczerwone w zakresie 760 – 1100 nm pochodzące od płomienia i w ten sposób umożliwia stwierdzenie jego obecności. Kąt wykrywania wynosi w przybliżeniu 60 stopni.
Podstawowe parametry:
Napięcie pracy 5 V przy sygnalizacji analogowej, 3,3 V – przy cyfrowej,
Oddzielne wyjścia analogowe i logiczne,
Regulowana czulość,
Wykrywane promieniowanie podczerwone w zakresie 760 – 1100 nm,
Wymiary 45 x 15 mm,
Masa 3 g.

Wyprowadzenia:
A0 – wyjście sygnału analogowego zależnego od natężenia promieniowania,
D0 – wyjście logiczne, 0 – przy braku bodźca, 1 – po przekroczeniu wartości progowej,
+ – zasilanie 5 V dla pomiarów analogowych, 3,3 V dla sygnalizacji logicznej,
G – masa.

Rys. 2.1.2. Połączenia z „Arduino”

Przykładowy program ilustruje sposób wykorzystania zarówno sygnału analogowego A0 jak i logicznego D0. Po przekroczeniu wartości progowej i wystąpieniu jedynki logicznej na wyjściu D0 zaświecana jest (znajdująca się na płytce) dioda elektroluminescencyjna połączona z nóżką 13. Próg czułości reguluje się za pomocą potencjometru.

```
int Led = 13; // wyjście dla diody świecacej
int buttonpin = 3; // wejście logiczne dla czujnika płomieni
int analoog = A3; // wejście analogowe dla czujnika
int val ;// deklaracja zmiennych
float sensor;
void setup ()
{
  pinMode (Led, OUTPUT);      // kontakt dla diody jako wyjście
  pinMode (buttonpin, INPUT); // wejście logiczne dla czujnika
  pinMode (analoog, INPUT);    // wejście analogowe dla czujnika
  Serial.begin(9600);
}

void loop ()
{
  sensor = analogRead(analoog);
  Serial.println(sensor); // wyświetlenie temperatury
  val = digitalRead (buttonpin) ;// użycie jako logicznego wejścia 3 do odczytu
  if (val == HIGH) // Wykrycie płomienia powoduje miganie diody
   {digitalWrite (Led, HIGH);
   }
  else
   {digitalWrite (Led, LOW);
   }
  delay(1000);
}
```

Do wykrywania płomieni świecy, kominka, ale także promieniowania słonecznego i innych źródeł można użyć także czujnika promieniowania ultrafioletowego. Szczególne dogodne do tego celu są czujniki zawierające fotodiodę i scalony z nią wzmacniacz. Pozwala to na podłączenie czujnika bezpośrednio do wejścia analogowego mikrokomputera i odczyt mierzonego napięcia jak w powyższym przykładzie. Jednym z czujników tego rodzaju jest „Tocon nano". Posiada on trzy wyprowadzenia: zasilanie 5V, masę i wyjście napięcia proporcjonalnego do oświetlenia. W odległości około 1 m od świecy napięcie na wyjściu wynosi około 25 mV. Przy oświetleniu 1 nW/cm^2 napięcie wyjściowe dochodzi do 280 mV. Charakterystyka widmowa czujnika pokrywa w przybliżeniu zakres 225 – 350 µm. W układzie wykrywacza płomieni przy świetle dziennym konieczne jest przysłonięcie czujnika specjalnym filtrem przepuszczającym tylko promieniowanie ultrafioletowe albo umieszczenie go w czarnej rurce skierowanej w kierunku płomienia dla zminimalizowania wpływu światła słonecznego. Fotodioda nie może być osłonięta szybkami ze zwykłych tworzyw sztucznych ponieważ tłumią one promieniowanie ultrafioletowe.

Termometr cyfrowy

Rys. 2.2.1. Moduł termometru cyfrowego SE017

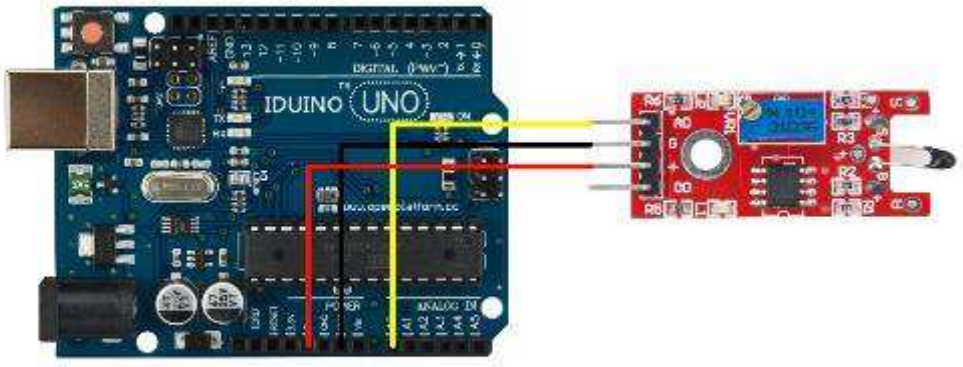
Wyposażony w termistor moduł SE017 posiada zarówno wyjście analogowe A0 jak i logiczne D0.
Podstawowe parametry:
Termistor NTC-MF52 3950
Zakres pomiarowy – -55ºC – +125ºC
Wyprowadzenia:
A0 – wyjście sygnału analogowego,
D0 – wyjście sygnału logicznego,
G – masa,
+ – zasilanie (napięcie odniesienia) 5 V.

Temperatura obliczana jest ze wzoru Steinhart-Harta:
$1/T = A + B \ln(R) + C[\ln(R)]^3$
Gdzie R [Ω] jest opornością termistora w danej temperaturze T [ºK], a A, B i C współczynnikami zależnymi od typu termistora i zakresu pomiarowego, a także w pewnym stopniu od temperatury otoczenia. W postaci ogólnej wzór zawiera również człon kwadratowy, ale jest on tutaj pomijalnie mały.
Wartości współczynników:
A = 0,001129148,
B = 0,000234125,
C = 0,0000000876741.
W przypadku gdy współczynniki te dają wyniki niedokładne można obliczyć ich wartości posługując się dostępnym w Internecie kalkulatorem dla termistorów.

Przykładowy program odczytu temperatury

```
#include <math.h>
double Thermister(int RawADC) {
double Temp;
Temp = log(((10240000/RawADC) - 10000));
Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))*
Temp );
Temp = Temp - 273.15;
return Temp;
}
```

```
void setup() {
  Serial.begin(9600);
}
void loop()
{ Serial.print(Thermister(analogRead(0)));
  Serial.println("c");
  delay(1000); }
```



Rys. 2.2.2. Połączenia z „Arduino”

Fotoopornik

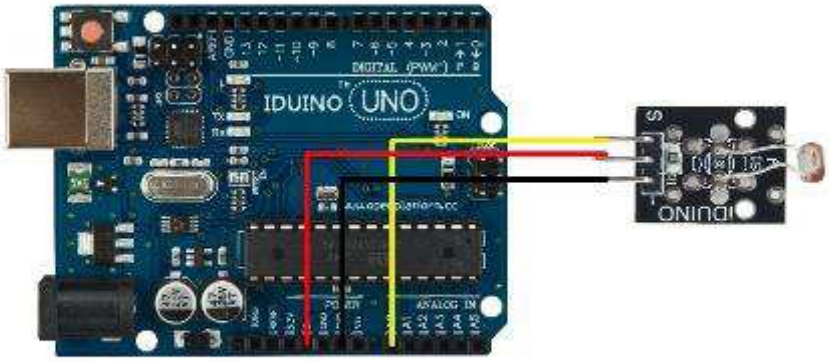
Rys. 2.3.1. Moduł fotoopornika

Czujnik służy do pomiaru natężenia padającego światła.
Parametry:
Napięcie zasilania 5 V,
Wymiary płytki drukowanej 28 x 15 mm,
Masa 2 g.
Wyprowadzenia:
S – wyjście sygnału analogowego,
+ – zasilanie,
Masa.

Przykładowy program odczytu wartości mierzonej i jej wyświetlania przez zmianę okresu migania diody elektroluminescencyjnej:

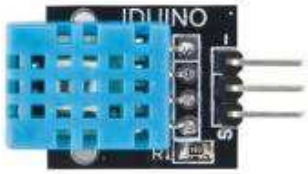
```
int sensorPin = A5; // deklaracja wejścia analogowego dla potencjometru
int ledPin = 13; // deklaracja wyjścia diody świecącej
int sensorValue = 0; // zmienna dla odczytanej z czujnika wartości
void setup() {
  pinMode(ledPin, OUTPUT);
```

```
  Serial.begin(9600);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  digitalWrite(ledPin, HIGH);
  delay(sensorValue);
  digitalWrite(ledPin, LOW);
  delay(sensorValue);
  Serial.println(sensorValue, DEC);
}
```



Rys. 2.3.2. Połączenia z „Arduino”

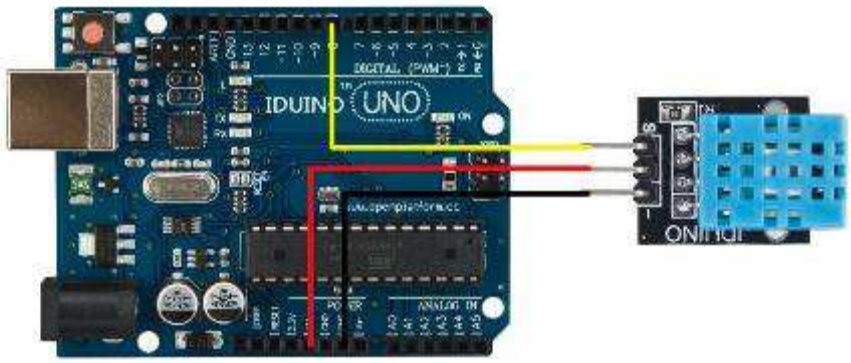
Termometr z wilgociomierzem



Rys. 2.4.1. Moduł termometru z wilgociomierzem

Moduł SE052 jest wyposażony w cyfrowy czujnik DHT11. Dostarcza on wykalibrowanego cyfrowego sygnału odpowiadającego zmierzonej temperaturze i wilgotności. Do pomiaru wilgotności służy

element oporowy, a do pomiaru temperatury – termistor, przy czym oba są połączone z 8-bitowym mikroprocesorem przetwarzającym ich dane.



Rys. 2.4.2. Połączenia z „Arduino”

Parametry:
Model – DHT11,
Napięcie zasilania – 5V (3,5 – 5,5 V),
Zakres pomiaru temperatury – 0 – 50 ºC,
Zakres pomiaru wilgotności względnej – 20 – 90%,
Dokładność – +/- 0,2 ºC, +/- 5%,
Rozdzielczość 16 bitów,
Dane w postaci 5 bajtów, kolejno: część całkowita wilgotności względnej, część ułamkowa wilgotności, część całkowita temperatury, część ułamkowa temperatury, 8 niższych bitów sumy kontrolnej, bajty 2 i 4 mają zawsze wartość 0,
Pobór prądu 0,3 mA,
Prąd spoczynkowy 60 µA,
Wymiary 12 x 15,5 x 5,5 mm,
Zasięg transmisji danych czujnika DHT11 przy zasilaniu 5 V – 20 m, 3,5 V – 0,2 m.
Wyprowadzenia:
S – wyjście sygnału pomiarowego,
- – masa,
+ – zasilanie 5 V.

Przykładowy program:

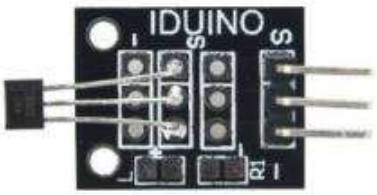
```
int DHpin = 8;
byte dat [5];
byte read_data () {
  byte data;
  for (int i = 0; i < 8; i ++) {
    if (digitalRead (DHpin) == LOW) {
      while (digitalRead (DHpin) == LOW); // opóźnienie 50 us
      delayMicroseconds (30); // opóźnienie dla analizy stanu logicznego
      if (digitalRead (DHpin) == HIGH)
```

```
    data |= (1 << (7-i)); // odczyt od najwyższych do najniższych bitów
    while (digitalRead (DHpin) == HIGH); // przy '1 ' opóźnienie
   }
  }
  return data;
}
void start_test () {
  digitalWrite (DHpin, LOW); // wydanie zera – sygnału startowego
  delay (30); // opóźnienie powyżej 18 ms dla rozpoznania sygnału startowego DHT11
  digitalWrite (DHpin, HIGH);
  delayMicroseconds (40); // Oczekiwanie na odpowiedź DHT11
  pinMode (DHpin, INPUT);
  while (digitalRead (DHpin) == HIGH);
  delayMicroseconds (80); // odpowiedź DHT11, odczekanie 80 us
  if (digitalRead (DHpin) == LOW);
  delayMicroseconds (80); // DHT11 80 us po starcie
  sending data
  for (int i = 0; i < 4; i ++) // odczyt temperatury i wilgotności, bit parzystości ignorowany
    dat[i] = read_data ();
  pinMode (DHpin, OUTPUT);
  digitalWrite (DHpin, HIGH); // nadanie danych po zwolnieniu magistrali,
  // oczekiwanie na następny start
}

void setup () {
  Serial.begin (9600);
  pinMode (DHpin, OUTPUT);
}
void loop () {
  start_test ();
  Serial.print ("Current humdity =");
  Serial.print (dat [0], DEC); // wyświetlenie części całkowitej wilgotności;
  Serial.print ('.');
  Serial.print (dat [1], DEC); // wyświetlenie części ułamkowej wilgotności;
  Serial.println ('%');
  Serial.print ("Current temperature =");
  Serial.print (dat [2], DEC); // wyświetlenie części ułamkowej temperatury;
  Serial.print ('.');
  Serial.print (dat [3], DEC); // wyświetlenie części ułamkowej temperatury;
  Serial.println ('C');
  delay (700);
}
```

W pozycji [110] proponowany jest m.in. następujący format pakietu telemetrycznego dla transmisji danych w systemie „LoRa” (przystosowany do możliwości DHT11):

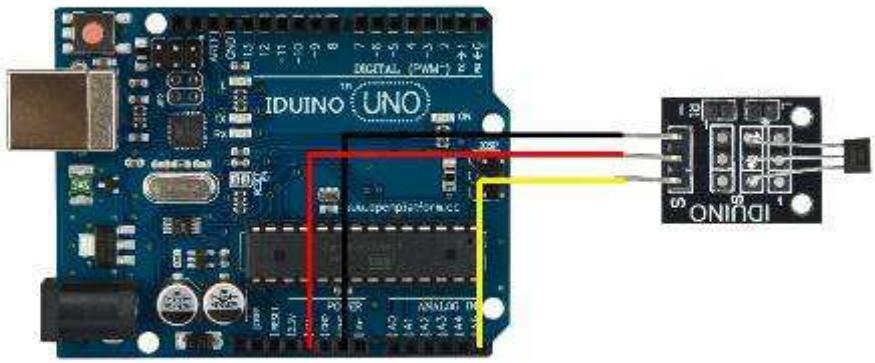
- Poz. 0 – litera „T” dla pakietów telemetrycznych tego rodzaju, inne litery dla pakietów zawierających inne typy danych,
- Poz. 1–8 – wilgotność,
- Poz. 9–16 – temperatura,
- Poz. 17–24 – ciśnienie atmosferyczne,
- Poz. 25 – dodatkowa sygnalizacja alarmu itp.
- Poz. 26–39 – do dowolnego użytku.

Czujnik natężenia pola magnetycznego z halotronem



Rys. 2.5.1. Moduł SE022

Czujnik SE022 dostarcza analogowego napięcia zależnego od zmierzonego przez halotron natężenia pola magnetycznego.
Parametry:
Napięcie zasilania – 5 V,
Wymiary płytki – 28 x 15 mm,
Masa – 2 g.
Wyprowadzenia:
S – analogowy sygnał pomiarowy
+ – zasilanie,
- – masa.

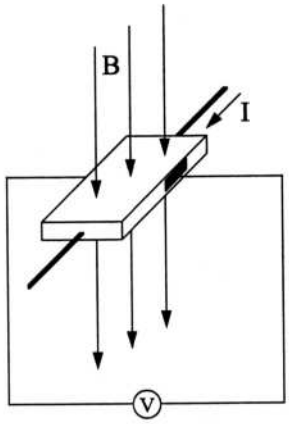


Rys. 2.5.2. Połączenia z „Arduino”

Przykładowy program:

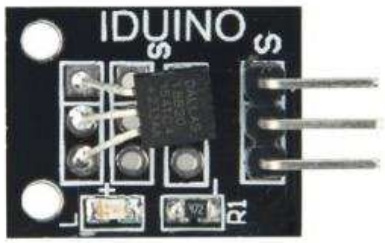
```
int sensorPin = A5; // deklaracja wejścia analogowego
int ledPin = 13; // deklaracja wyjścia dla diody
int sensorValue = 0; // zmienna dla odczytanej wartości
void setup () {
  pinMode (ledPin, OUTPUT);
  Serial.begin (9600);
}
void loop () {
  sensorValue = analogRead (sensorPin);
  digitalWrite (ledPin, HIGH);
  delay (sensorValue);
  digitalWrite (ledPin, LOW);
  delay (sensorValue);
  Serial.println (sensorValue, DEC);
}
```

Halotron jest półprzewodnikowym podzespołem służącym do pomiaru stałych pól magnetycznych. Składa się z cienkiej płytki półprzewodnikowej przez którą w trakcie pracy płynie prąd o stałym natężeniu. Linie sił pola magnetycznego przechodzące przez płytkę prostopadle do kierunku przepływu prądu powodują przesunięcie się ładunków elektrycznych (czyli przepływającego prądu) na jedną stronę płytki. Na ściankach prostopadłych do kierunków prądu i pola magnetycznego powstaje napięcie o wartości i polaryzacji zależnych od natężenia i kierunku pola magnetycznego (rys. 2.5.3).
Dla dokonania pomiaru natężenia otaczającego pola magnetycznego (np. ziemskiego) należy obracać czujnik aż do otrzymania maksymalnej wartości napięcia wyjściowego. W przypadku urządzeń, w których kierunek linii pola jest znany wystarczy tylko odpowiednio umieścić halotron.



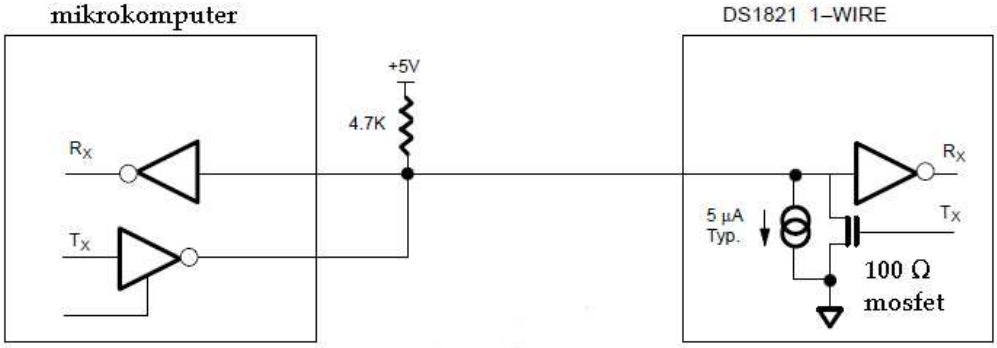
Rys. 2.5.3. Zasada działania halotronu

Termometr z czujnikiem DS18B20

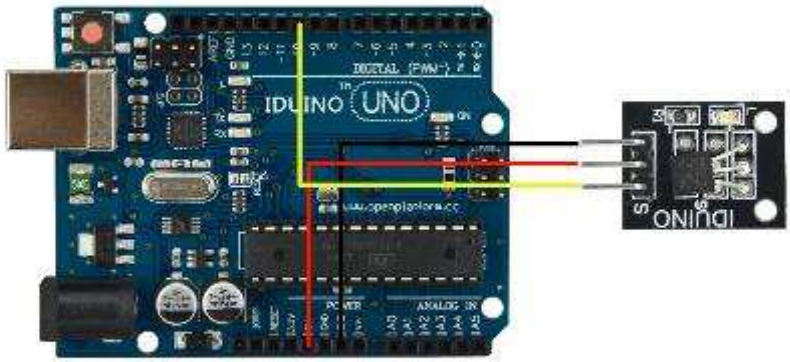


Rys. 2.6.1. Moduł SE029

Parametry modułu termometru:
Czujnik DS18B20,
Zakres pomiarowy – -55 ºC – +125 ºC
Dokładność – +/- 0,5 ºC,
Zasilanie – 5 V.
Wyprowadzenia:
S – wyjście sygnału,
+ – kontakt środkowy, zasilanie,
- – masa.



Rys. 2.6.2. Złącze „1-Wire”. Nazwa nawiązuje do pojedynczego przewodu sygnałowego i nie uwzględnia połączenia masy. Analogiczne tworzone są nazwy dla złączy dwu- lub trzy przewodowych



Rys. 2.6.3. Połączenia z „Arduino”

Przykładowy program (wymaga biblioteki *OneWire*):

```
#include <OneWire.h>
// We./wy. dla termometru DS18S20
OneWire ds(10); // na wyprowadzeniu 10
void setup(void) {
  // inicjalizacja wejść i wyjść
  // uruchomienie złącza szeregowego
  Serial.begin(9600);
}
void loop(void) {
  // Dla przeliczenia danych wejściowych na stopnie C
  int HighByte, LowByte, TReading, SignBit, Tc_100, Whole, Fract;
  byte i;
  byte present = 0;
  byte data[12];
```

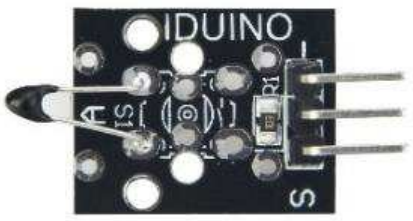
```
byte addr[8];
if ( !ds.search(addr)) {
  Serial.print("No more addresses.\n");
  ds.reset_search();
  return;
}
Serial.print("R=");
for( i = 0; i < 8; i++) {
  Serial.print(addr[i], HEX);
  Serial.print(" ");
}
if ( OneWire::crc8( addr, 7) != addr[7]) {
  Serial.print("CRC is not valid!\n");
  return;
}
if ( addr[0] == 0x10) {
  Serial.print("Device is a DS18S20 family device.\n");
}
else if ( addr[0] == 0x28) {
  Serial.print("Device is a DS18B20 family device.\n");
}
else {
  Serial.print("Device family is not recognized: 0x");
  Serial.println(addr[0],HEX);
  return;
}
ds.reset();
ds.select(addr);
ds.write(0x44,1); // początek konwersji, czujnik zasilany z Arduino
delay(1000); // opóźnienie 750 ms może wystarczyć lub nie
// można użyć ds.depower(), ale stan pozostaje po ponownym starcie.
present = ds.reset();
ds.select(addr);
ds.write(0xBE); // Odczyt danych z pamięci podręcznej
Serial.print("P=");
Serial.print(present,HEX);
Serial.print(" ");
for ( i = 0; i < 9; i++) { // odbierane 9 bajtów
  data[i] = ds.read();
  Serial.print(data[i], HEX);
  Serial.print(" ");
}
Serial.print(" CRC=");
Serial.print( OneWire::crc8( data, 8), HEX);
Serial.println();
//Conversion of raw data to C
LowByte = data[0];
HighByte = data[1];
TReading = (HighByte << 8) + LowByte;
SignBit = TReading & 0x8000; // sprawdzanie najwyższego bitu
if (SignBit) // wartość ujemna
{TReading = (TReading ^ 0xffff) + 1; // uzupełnienie dwójkowe
}
Tc_100 = (6 * TReading) + TReading / 4; // mnożenie przez (100 * 0.0625) or 6.25
Whole = Tc_100 / 100; // oddzielenie części całkowitej od ułamkowej
```

```
  Fract = Tc_100 % 100;
  if (SignBit) // jeżeli wartość ujemna
  {Serial.print("-");
  }
  Serial.print(Whole);
  Serial.print(".");
  if (Fract < 10)
  {Serial.print("0");
  }
  Serial.print(Fract);
  Serial.print("\n");
  // koniec przeliczania na stopnie C
}
```

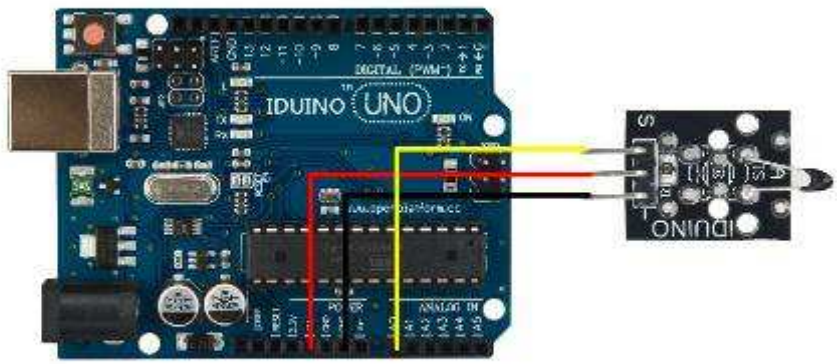
Termometr analogowy ST1147

Parametry:
Termistor – NTC-MF52 3950,
Zakres pomiarowy – -55 ºC – +125 ºC,
Dokładność – +/- 0,5 ºC,
Opornik polaryzujący 10 kΩ.
Wyprowadzenia:
S – wyjście sygnałowe,
- – masa,
+ – zasilanie 5 V.

Temperatura obliczana jest ze wzoru Steinhart-Harta:
$1/T = A + B \ln(R) + C[\ln(R)]^3$
Gdzie R [Ω] jest opornością termistora w danej temperaturze T [ºK], a A, B i C współczynnikami zależnymi od typu termistora i zakresu pomiarowego, a także w pewnym stopniu od temperatury otoczenia. W postaci ogólnej wzór zawiera również człon kwadratowy, ale jest on tutaj pomijalnie mały.
Wartości współczynników:
A = 0,001129148,
B = 0,000234125,
C = 0,0000000876741.
W przypadku gdy współczynniki te dają wyniki niedokładne można obliczyć ich wartości posługując się dostępnym w Internecie kalkulatorem dla termistorów.



Rys. 2.7.1. Moduł termometru termistorowego



Rys. 2.7.2. Sposób podłączenia do „Arduino”

Przykładowy program:

```
#include <math.h>
double Thermister(int RawADC) {
double Temp;
  Temp = log(((10240000/RawADC) - 10000));
  Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))*
  Temp );
  Temp = Temp - 273.15;
  return Temp;
}
void setup() {
  Serial.begin(9600);
}
void loop()
  { Serial.print(Thermister(analogRead(0)));
  Serial.println("c");
  delay(1000); }
```

Czujnik barometryczny BME280

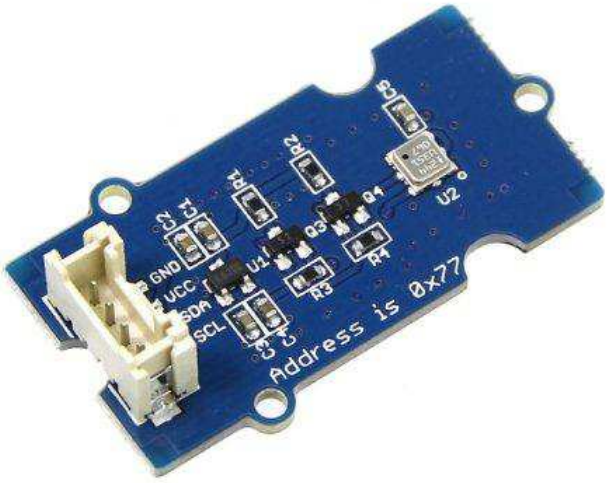
BME280 firmy „Bosch” jest trójkanałowym czujnikiem służącym do pomiaru wilgotności względnej powietrza, temperatury i ciśnienia atmosferycznego. Do połączenia z mikrokomputerem służy magistrala I2C lub SPI.
Parametry:
Tolerancja pomiaru wilgotności względnej 3%,
Czas odpowiedzi – 1 sekunda,
Histereza <= 2%,
Zakres pomiaru ciśnienia – 300 – 2100 hPa, odpow. +9000 m do -500 m w stosunku do poziomu morza,
Dokładność względna +/- 0,12 hPa, odpowiada +/- 1 m,
Dokładność bezwzględna +/- 1 hPa (w zakresie 950 – 1050 hPa przy temperaturach 0 – 40 ºC, .

Zakres pomiaru temperatury - -40 - +85 ºC (pełna dokładność w zakresie 0 - +65 ºC),

Fot. 2.8.1. BME280

Czujnik barometryczny BMP180

BMP180 firmy „Grove” jest dwukanałowym precyzyjnym czujnikiem służącym do pomiaru ciśnienia atmosferycznego i temperatury.
Parametry:
Pomiar ciśnienia w zakresie 300 – 1100 hPa,
Dokładność pomiaru ciśnienia – 0,02 hPa,
Zakres pomiaru temperatury – -40 - +85 ºC,
Dokładność pomiaru temperatury +/- 0,1 ºC,
Zakres pomiaru wysokości -500 – _9000 m n.p.m.,
Dokładność pomiaru wysokości 0,5 m lub 0,17 m w zależności od trybu pomiaru,
Złącze I2C, adres na złączu 0x77,
Napięcie zasilania 3,3 V lub 5 V,
Pobór prądu – maks. 20 µA,
Wymiary płytki 40,1 x 20,2 z 9,7 mm.



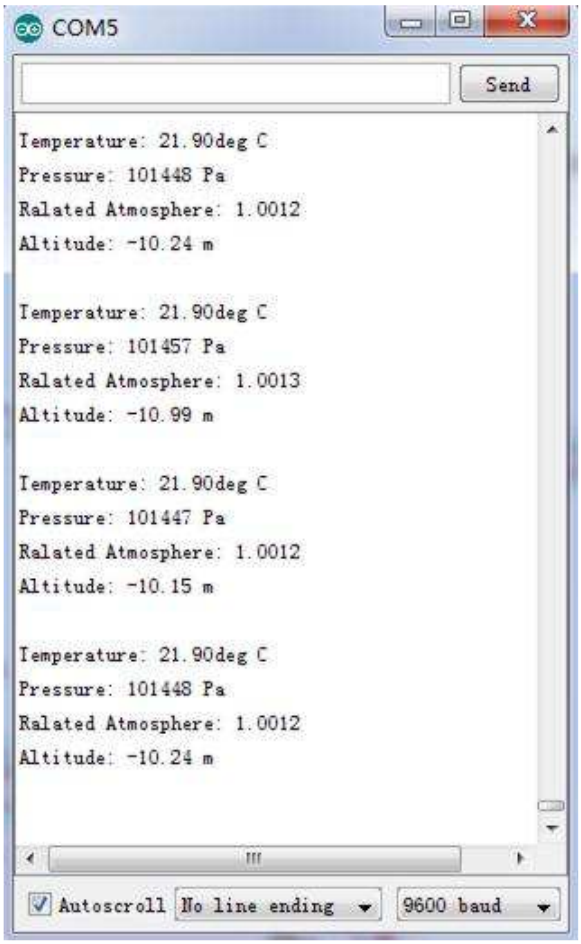
Fot. 2.9.1. BMP180

Przykładowy program odczytu danych dla „Arduino” (źródło *wiki.seed.cc*):

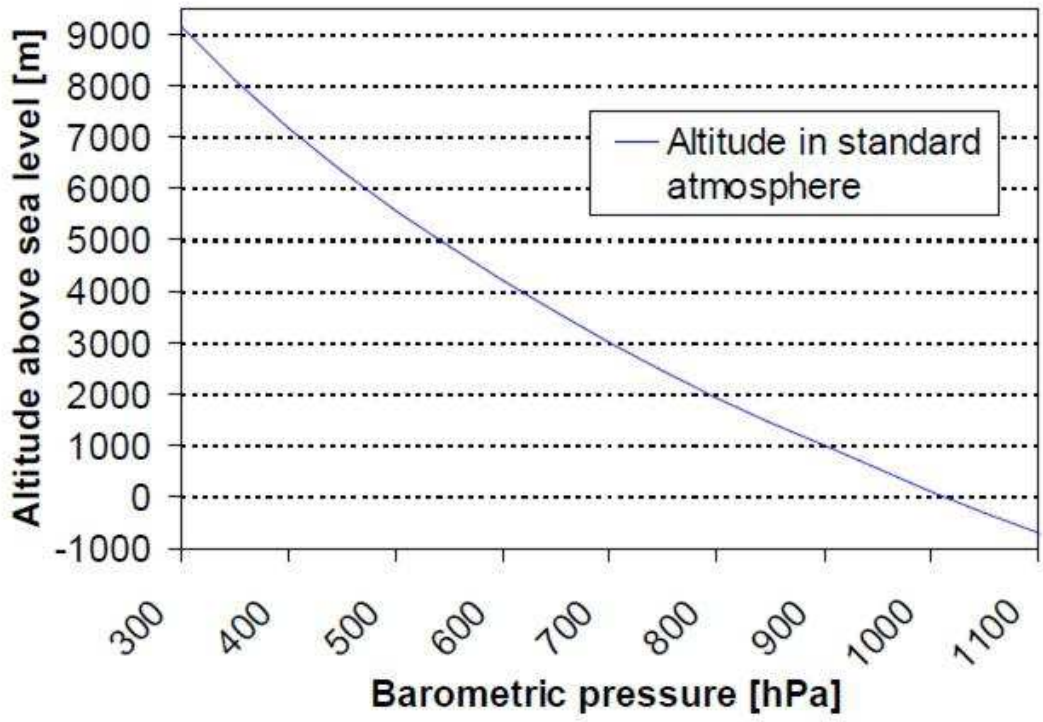
```
/* Program demonstracyjny dla barometru
   * Oparty w znacznej części na kodzie Jima Lindbloma
   * Odczyt ciśnienia, wysokości i temperatury z BMP085.
   * Serial.print wydaje z przepływnością 9600 bodów do okna monitora szeregoweg.
```

```
 *
 * Autorstwo:http://www.seeedstudio.com
 */
#include "Barometer.h"
#include <Wire.h>
float temperature;
float pressure;
float atm;
float altitude;
Barometer myBarometer;
void setup(){
  Serial.begin(9600);
  myBarometer.init();
}

void loop()
{
    temperature = myBarometer.bmp085GetTemperature(myBarometer.bmp085ReadUT()); // Odczyt temperatury, funkcja bmp085ReadUT musi być najpierw wywołana
    pressure = myBarometer.bmp085GetPressure(myBarometer.bmp085ReadUP());//Odczyt ciśnienia
    altitude = myBarometer.calcAltitude(pressure); //Wysokość w m nie skompensowana
    atm = pressure / 101325;

  Serial.print("Temperature: ");
  Serial.print(temperature, 2); //dwa miejsca dziesiętne
  Serial.println("deg C");

  Serial.print("Pressure: ");
  Serial.print(pressure, 0); //tylko liczby całkowitewhole number only.
  Serial.println(" Pa");

  Serial.print("Ralated Atmosphere: ");
  Serial.println(atm, 4); //cztery miejsca dziesiętne

  Serial.print("Altitude: ");
  Serial.print(altitude, 2); //dwa miejsca dziesiętne
  Serial.println(" m");

  Serial.println();

  delay(1000); //odczekanie sekundy przed następnym odczytem.
}
```

Program wymaga zainstalowania biblioteki *Grove_Barometer_Sensor*.



Rys. 2.9.2. Dane wyświetlane w konsoli szeregowej



Rys. 2.9.3. Związek między wysokością n.p.m w metrach i ciśnieniem atmosferycznym w hPa dla standardowej atmosfery

Przykładowy program odczytu danych dla „Maliny”:

```
#!/usr/bin/python
   import smbus
   import RPi.GPIO as GPIO
   #import grovepi
   from grove_i2c_barometic_sensor import BMP085
   #
================================================================
   #  Przykładowy kod programu
   #
================================================================
   # Inicjalizacja BMP085 w domyślnym trybie STANDARD
   # bmp = BMP085(0x77, debug=True)
   bmp = BMP085(0x77, 1)

   # Dla innych trybów usunąć znak komentarza z wybranej linii:
   # bmp = BMP085(0x77, 0)  # tryb ULTRALOWPOWER
   # bmp = BMP085(0x77, 1)  # tryb STANDARD
   # bmp = BMP085(0x77, 2)  # tryb HIRES
   # bmp = BMP085(0x77, 3)  # tryb ULTRAHIRES

   rev = GPIO.RPI_REVISION
   if rev == 2 or rev == 3:
      bus = smbus.SMBus(1)
   else:
      bus = smbus.SMBus(0)
   temp = bmp.readTemperature()
   # Odczyt bieżącego ciśnienia
   pressure = bmp.readPressure()
   # Dla obliczenia wysokości w oparciu o średnie ciśnienie na poziomie morza
   # (1013.25 hPa) należy wywołać funkcje jak podano, ale nie jest to zbyt dokładne
   # altitude = bmp.readAltitude()
   # Dla otrzymania wyniku dokładniejszego należy podać rzeczywiste ciśnienie na poziomie morza
   # przykładowo dla aktualnego ciśnienia 1023.50 hPa
   # należy wprowadzić 102350 ponieważ w liczbie tej zawarte są dwa miejsca dziesiętne
   altitude = bmp.readAltitude(101560)
   print "Temperature: %.2f C" % temp
   print "Pressure:    %.2f hPa" % (pressure / 100.0)
   print "Altitude:    %.2f m" % altitude
```

W celu ewentualnego dopasowania programu do własnych potrzeb należy przejść do jego katalogu:
cd yourpath/GrovePi/Software/Python/grove_barometer/adafruit

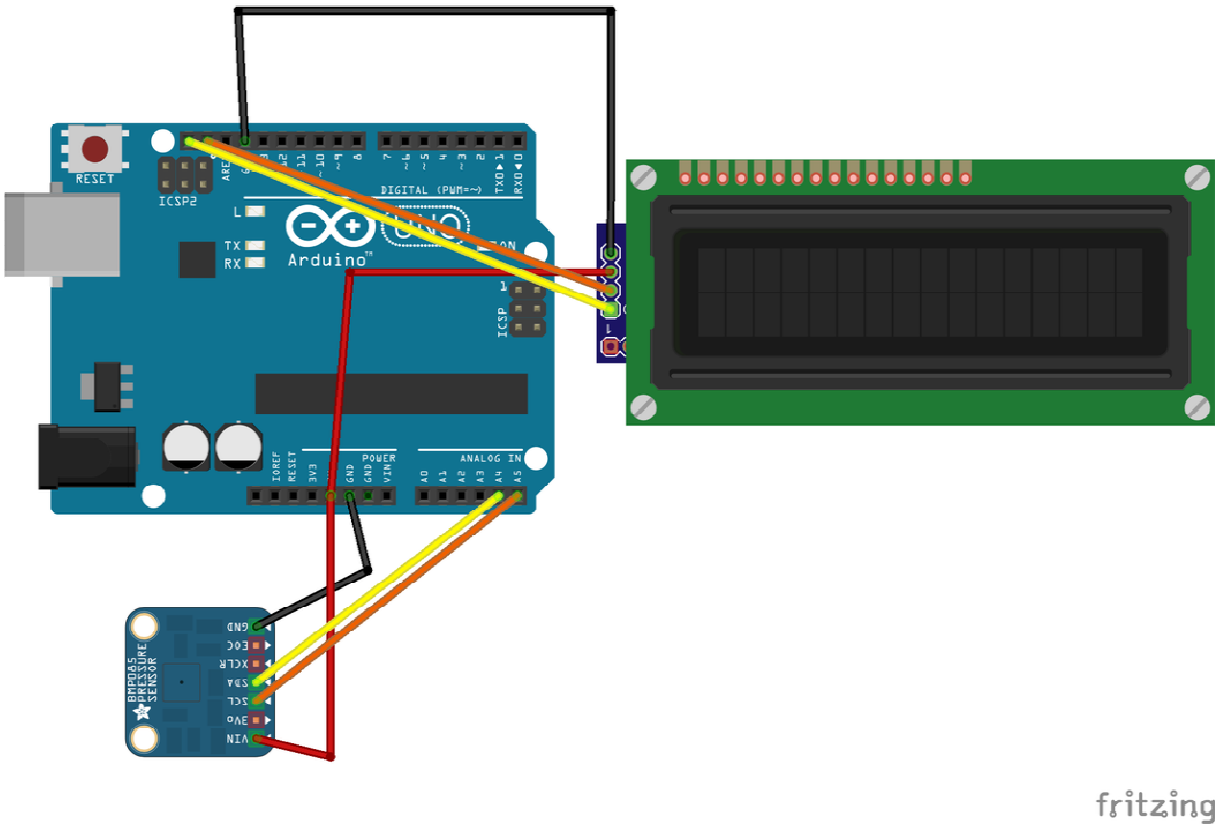
i wywołać edytor nano
*nano grove_i2c_barometic_sensor_example.**py* *# "Ctrl+x" **to exit** #*

natomiast do jego wywołania służy polecenie
*sudo **python** grove_i2c_barometic_sensor_example.**py***

```
pi@raspberrypi: ~/software/GrovePi/Software/Python/grove_barometer/adafruit
pi@raspberrypi ~/software/GrovePi/Software/Python/grove_barometer/adafruit $ ls
Adafruit_I2C.py                      grove_i2c_barometic_sensor.py
Adafruit_I2C.pyc                     grove_i2c_barometic_sensor.pyc
grove_i2c_barometic_sensor_example.py
pi@raspberrypi ~/software/GrovePi/Software/Python/grove_barometer/adafruit $ sudo
 python grove_i2c_barometic_sensor_example.py
Temperature: 27.40 C
Pressure:    1006.15 hPa
Altitude:    95.96 m
pi@raspberrypi ~/software/GrovePi/Software/Python/grove_barometer/adafruit $
```

Rys. 2.9.4. Dane wyświetlane przez program

Przykładowy program odczytujący ciśnienie i wyświetlający je na wyświetlaczu ciekłokrystalicznym. Konieczne jest zainstalowanie bibliotek *liquidcrystal_i2c* i *bmp180*.



Rys. 2.9.5. Schemat połączeń

```
/*#########################################################################
 PLIK: Arduino_barometro_BMP_180
 WERSJA: 1.0
 Data ostaniej modyfikacji 01/07/2016
 Autor Fancello Salvatore
 Witryna http://progettiarduino.weebly.com
 Przykład do dowolnego wykorzystania.
 ######################################################################### */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <BMP180.h>
LiquidCrystal_I2C lcd(0x27,16,2);

BMP180 barometer;

int indicatorLed = 13;
// Wprowadzić dane dla własnej lokalizacji
float seaLevelPressure = 101325;
void setup()
{

  Serial.begin(9600);

  Wire.begin();

  pinMode(indicatorLed, OUTPUT);

  barometer = BMP180();

  if(barometer.EnsureConnected())
  {
    Serial.println("Connesso con BMP180.");
    digitalWrite(indicatorLed, HIGH); // Poziom wysoki na wyjściu diody sygnalizacyjnej.

    barometer.SoftReset();

    barometer.Initialize();
  }
  else
  {
    Serial.println("Sensore non coll");
    digitalWrite(indicatorLed, LOW);
  }
  lcd.backlight();
  lcd.setCursor(3, 0);
  lcd.print("Test BMP180");
  lcd.setCursor(0, 1);
  lcd.print("progettiarduino.weebly.com");
  delay(5000);
}
void loop()
{
  if(barometer.IsConnected)
  {
```

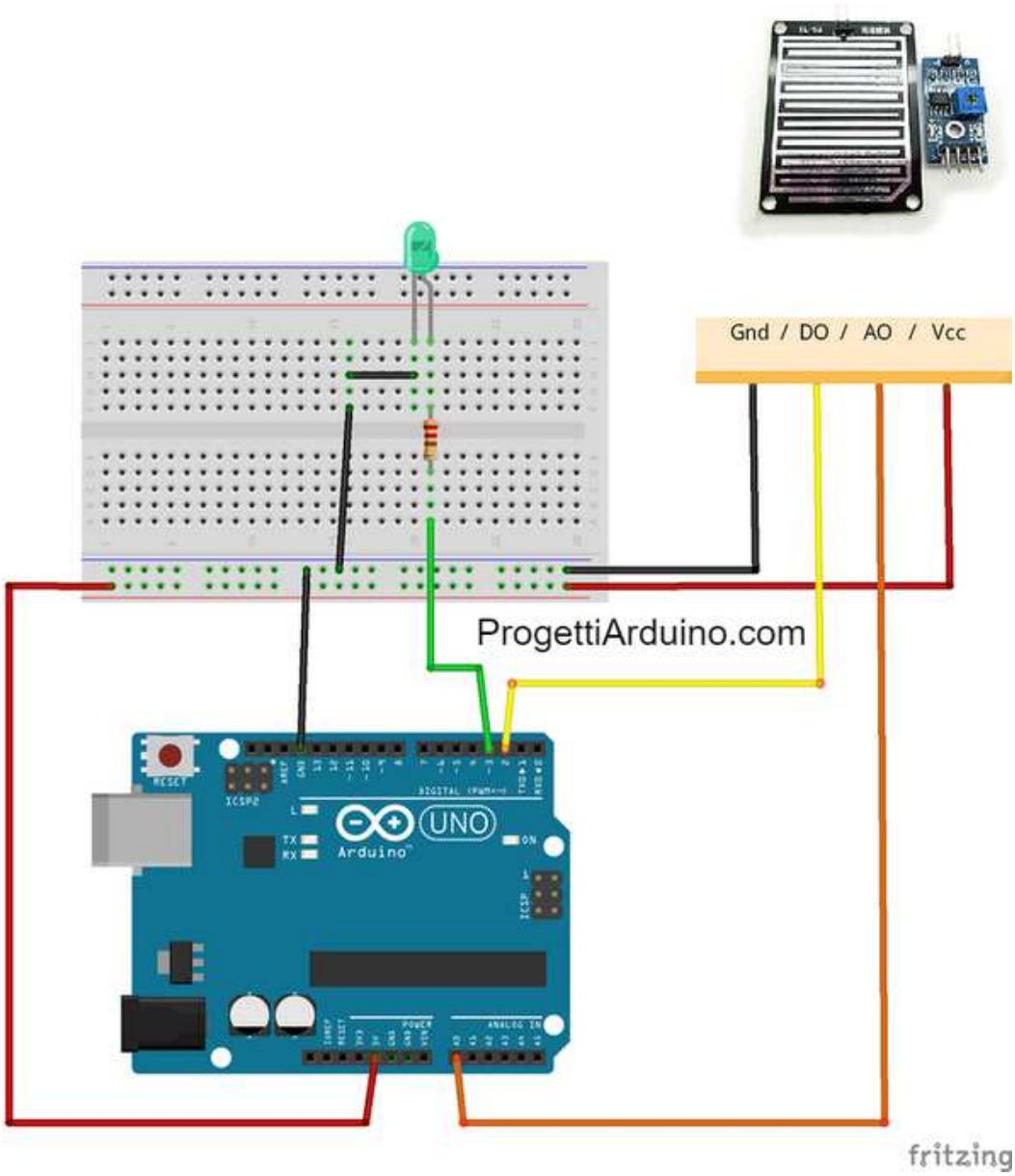
```
    long currentPressure = barometer.GetPressure();

    float altitude = barometer.GetAltitude(seaLevelPressure);

    float currentTemperature = barometer.GetTemperature();
    Serial.println();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(currentPressure);
    lcd.print(" Pa");
    lcd.setCursor(0, 1);
    lcd.print (altitude);
    lcd.print (" m");
    lcd.setCursor(9, 1);
    lcd.print (currentTemperature);
    lcd.print (" C");
    delay(1000);
  }
}
```

Czujnik opadów

Przykładowy program odczytuje stan wyjścia logicznego czujnika i w przypadku stwierdzenia opadu zaświeca diodę elektroluninescencyjną, a oprócz tego odczytuje wartość mierzoną z wyjścia analogowego informująca o intensywności opadu. W szereg z diodą włączony jest opornik 220 Ω.

```
// ITA: Assegno i pin ai led
// POL: Deklaracja wyjścia dla diody i wejścia dla czujnika
int pinIngresso=2;
int pinLed=3;

void setup()
{
   // POL: Włączenie złącza szeregowego (przepływność 9600 bodów)
   // ITA: Abilito la porta seriale (9600 baud)
  Serial.begin(9600);

  // POL: Przełączenie wyprowadzeń na potrzebne wejścia i wyjścia
  // ITA: Imposto i pin come INPUT e come OUTPUT
  pinMode(pinIngresso, INPUT);
  pinMode(pinLed,OUTPUT);
}

//POL: Deklaracje zmiennych
//ITA: Dichiarazione variabili

int pioggia=0;
int misurazione=0;

void loop()
{
  // ITA: Leggo l'ingresso digitale
  // POL: Odczyt wejścia logicznego
  pioggia = digitalRead(pinIngresso);
```

```
  if(pioggia==1)
    digitalWrite(pinLed,LOW);
  if(pioggia==0)
    digitalWrite(pinLed,HIGH);

    // ITA: Leggo l'ingresso analogico
    // POL: Odczyt wejścia analogowego
    misurazione = analogRead(A0);
    Serial.println(misurazione);
  delay(1000);
}
```

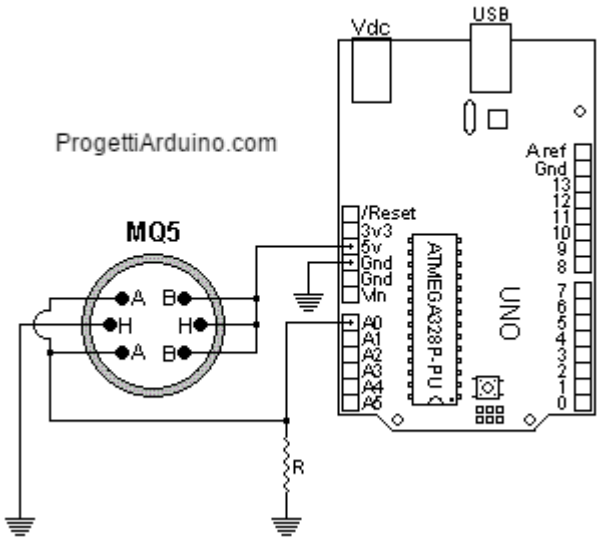


Rys. 2.10.1. Schemat połączeń czujnika opadów (źródło: *www.progettiarduino.com*)

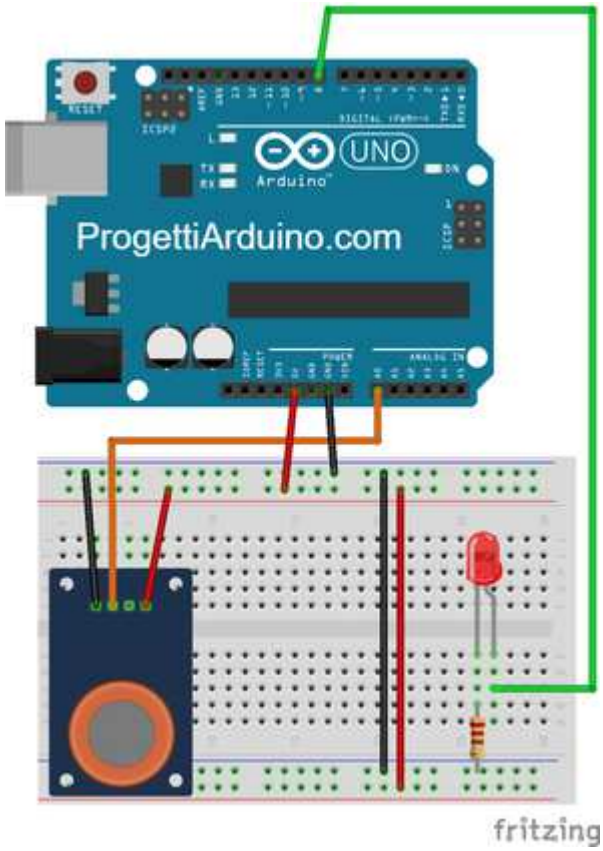
Wykrywacze gazów

Zastosowany w pierwszym wykrywaczu czujnik gazu typu MQ5 reaguje na obecność gazu ziemnego i miejskiego w powietrzu.

Rys. 2.11.1a Czujnik MQ5



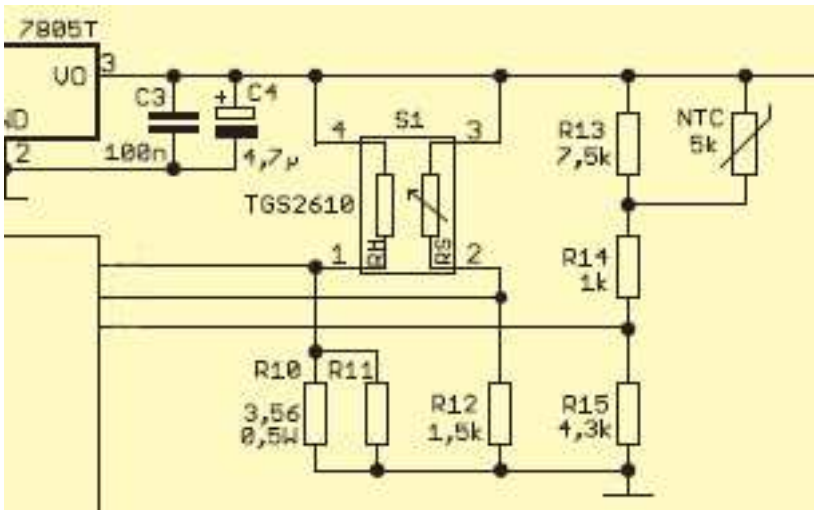
Rys. 2.11.1b Schemat podłączenia do „Arduino”



Rys. 2.11.2. Układ eksperymentalny. Obecność gazu jest sygnalizowana za pomocą diody elektroluminescencyjnej. W szereg z diodą włączony jest opornik 220 Ω (źródło: *www.progettiarduino.com*)

Przykładowy program:

```
// Wejście logiczne 8 nosi nazwę 'pin8'
int pin8 = 8;
// Wejście analogowe 0 nosi nazwę 'sensor'
int sensor = A0;
// Inicjalizacja sensorValue na 0
int sensorValue = 0;

// Funkcja setup() wywoływana raz po uruchomieniu programu
void setup() {
  // Inicjalizacja wyjścia 8
  pinMode(pin8, OUTPUT);
  // Inicjalizacja złącza szeregowego na 9600 bit/s
  Serial.begin(9600);
}

// Funkcja loop() pracuje bez końca
void loop() {
  // Odczyt wejścia analogowego 0 pod nazwą 'sensor'
  sensorValue = analogRead(sensor);
  // Wyświetlenie odczytanej wartości
  Serial.println(sensorValue, DEC);
  // jeśli sensorValue przekracza 500
  if (sensorValue > 500) {
    // Wysoki poziom na wyjściu logicznym 8 – zaświecenie diody
    digitalWrite(pin8, HIGH);
  }
  else {
    // Niski poziom na wyjściu logicznym 8 – zgaszenie diody
    digitalWrite(pin8, LOW);
  }
}
```



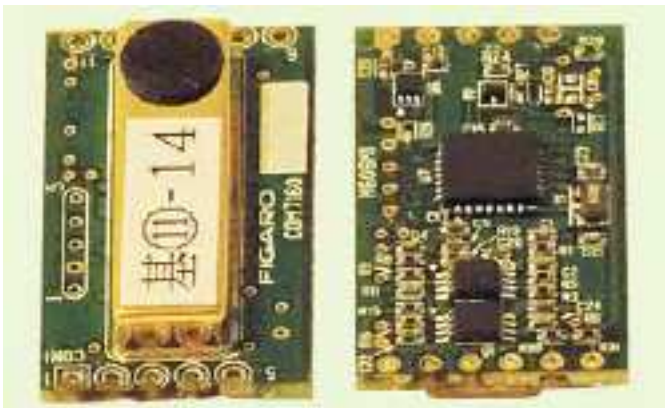
Rys. 2.11.3. Schemat połączeń dla czujnika TGS2610

Czujnik TGS2610 firmy „Figaro” reaguje natomiast na obecność wodoru, etanolu, metanu, izobutanu i propanu. Oparty o tlenek cynku element wymaga podgrzewania i zmienia swoją oporność w obecności wymienionych gazów i w zależności od ich stężenia w powietrzu, przy czym czułość zależy od ro-

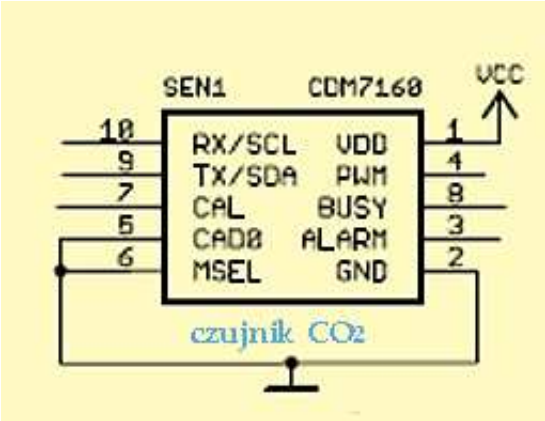
dzaju gazu. Typowym układem pomiarowym jest dzielnik napięcia złożony z opornika o tolerancji 1% (na schemacie R12) i TGS2610. Sygnał wyjściowy z dzielnika jest doprowadzany do wejścia analogowego mikrokomputera. Napięcie grzejnika musi leżeć w zakresie 4,8 – 5,2 V i być stabilizowane z dokładnością do 0,1 V aby nie dopuścić do jego przepalenia.
Wypadkowa oporność oporników R10 i R11 w obwodzie grzejnika (wyprowadzenia 1, 4) powinna wynosić 3,56 Ω z tolerancją +/- 1%, a napięcie na nich jest również doprowadzone do wejścia analogowego mikrokomputera. Grzejnik pobiera prąd 80 – 100 mA.
Czujnik jest zasadniczo przewidziany do ostrzegania o wybuchowej koncentracji gazów, ale bez dokładnej kalibracji układu ma on charakter doświadczalny i nie zapewnia dostatecznie godnej zaufania informacji dla urządzeń alarmowych. Opis układu pochodzi z nr 9/2012 miesięcznika „Funkamateur”.

Pomiar stężenia dwutlenku węgla

Objętościowa zawartość dwutlenku węgla w powietrzu wynosi w przybliżeniu 0,04 %. Wyższe koncentracje mogą spowodować złe samopoczucie i trudności w koncentracji, a powyżej pewnej granicy mogą odbijać się ujemnie na zdrowiu lub nawet doprowadzić do śmierci.
Normy DIN EN 13799 i DIN EN 15251 definiują dobrą jakość powietrza przy stężeniu dwutlenku węgla poniżej 0,08%, z zakresie 0,08 – 0,1% jakość powietrza jest uważana za średnią a w zakresie 0,1 – 0,14% za mierną, powyżej – za niską. Do 0,3 % stężenia dwutlenek wegla nie powoduje jeszcze żadnych szkód w organiźmie, w miejscu pracy dopuszczalne jest nawet stężenie 0,5% przy czasie pracy 8 godzin. Steżenia przekraczające 1,5% mogą wywołać już zakłócenia w pracy mózgu, a przy 8% śmierć może nastąpić w ciągu godziny. Pomiar stężenia dwutlenku wegla ma więc istotne znaczenie dla zdrowia i nawet życia ludzkiego, zwłaszcza w pomieszczeniach niedostatecznie wietrzonych, w atmosferze zanieczyszczonej spalinami itd.
W odróżnieniu od większości czujników opartych na specjalnie wybranych materiałach półprzewodnikowych i reagujących na obecność gazu na zasadzie elektrochemicznej. CDM7160 pracuje na zasadzie spektrografu w zakresie podczerwieni. Cząsteczki CO_2 pochłaniają promieniowanie podczerwone w paśmie 2400 nm (znacznie słabiej także w paśmie 3000 nm) i w tym właśnie podzakresie pracuje wymieniony czujnik. Posiada on wbudowany mikrokontroler i dostarcza danych pomiarowych poprzez złącze I2C lub SPI. Do wyboru złącza służy wejście MSEL, przy połączeniu z masą uruchamiane jest złącze I2C. Wejście CAD0 umożliwia zmianę ostatniego bitu w adresie. W zależności od połączenia z masą lub napięciem zasilania wartość ta wynosi 0 lub 1, co pozwala na równoległe podłączenie dwóch czujników. Napięcie zasilania modułu wynosi 5 V(4,5 – 5,25 V), a pobór mocy w momencie pomiaru – 75 mW. W układzie z rys. 1.12.2 adres dla zapisu wynosi 0xD7, a dla odczytu – 0x D1 (szesnastkowo). Wejście CAL służy do wprowadzenia podstawowej wartości koncentracji i jest połączone z RxD „Arduino”. Na wyjściu PWM wartość zmierzona jest wydawana w postaci impulsów o modulowanej szerokości, a sygnał BUSY informuje o dokonywaniu pomiaru (wartość 1) lub gotowości danych do odczytu (wartość 0). Czujnik pracuje w zakresie temperatur 0 – 50 ºC i dokonuje pomiaru co 2 sekundy. Odstęp ten można zmieniać w zakresie 2 – 256 sekund. Zakres pomiarowy wynosi 0,03 – 5% stężenia CO_2. Czujnik jest przewidziany do zastosowań pomiarowych i obserwacji, ale nie do użytku w aparaturze alarmowej.



Fot. 1.12.1. Moduł CDM7160



Rys. 1.12.2. Wyprowadzenia CDM7160

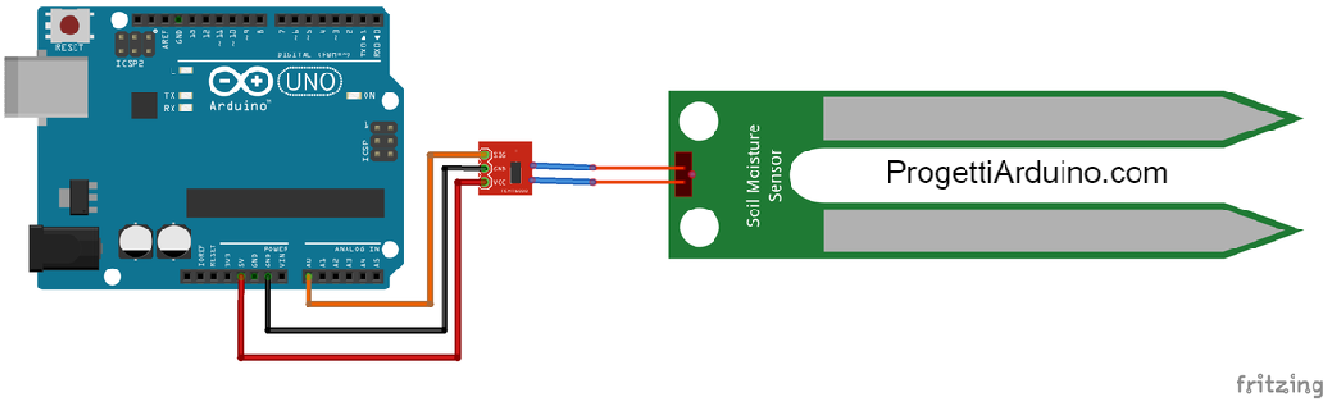
Opis układu autorstwa Klausa Sandera (*www.sander-electronic.de*) pochodzi z numeru 10/2016 „Funkamateura”.

Tabela 1.12.1. Wybrane typy czujników różnych rodzajów gazów

Typ czujnika	Wykrywane rodzaje gazów
KE25, KE50, SK25	Tlen
SP30, SP32	Alkohol
TGS2104	Spaliny benzyny
TGS2106	Spaliny oleju napędowego
TGS2180	Para wodna
TGS2210	Spaliny samochodowe, NOx + CO
TGS2442	Tlenek węgla
TGS2600, TGS2602	Ogólna jakość powietrza
TGS2610	Alkohol, metan, propan, izobutan
TGS2611, TGS842	Metan
TGS2612	Metan, propan, izopbutan
TGS3870	Metan i tlenek węgla
TGS4160, TGS4161	Dwutlenek węgla
TGS813	Gazy palne: alkohol, metan, propan, butan, wodór
TGS821	Wodór
TGS826	Amoniak

Fot. 1.12.3. Czujniki TGS2600, TGS2442 i TGS821

Czujnik wilgotności gruntu



Rys. 2.13.1. Czujnik wilgotności gruntu i jego sposób połączenia z „Arduino” (źródło: *www.progettiarduino.com*)

Czujnik mierzy oporność gruntu i jest zasilany napięciem 5 V z płytki mikrokomputera. Jego wyjściowe napięcie jest mierzone na wejściu analogowym A0.

Przykładowy program dla „Arduino”:

```
const int soglia_critica = 300; // Próg włączenia diody świecącej
void setup()
{
Serial.begin(9600);
pinMode(9, OUTPUT);
}
void loop()
{
  int sensorValue = analogRead(A0); // Odczyt wartości analogowej
  Serial.println(sensorValue); // Wyświetlenie wartości na ekranie
  if (sensorValue <= soglia_critica)
    digitalWrite(9,HIGH); // Zaświecenie diody po przekroczeniu progu
  else
    digitalWrite(9,LOW); // Zgaszenie diody
  delay(2000); // Przerwa 2 sekund
}
```

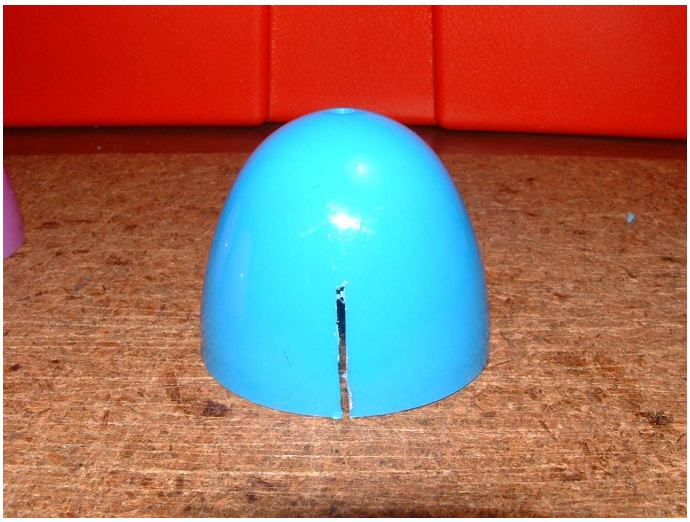
Czujniki własnej konstrukcji

Niektóre z czujników i związanych z nimi układów pomiarowych można konstruować samodzielnie co jest szczególnie interesujące w projektach dydaktycznych albo w przypadku rozwiązań nie spotykanych na rynku lub dostępnych tylko po cenach przekraczających możliwości finansowe prywatnych użytkowników. Dobrym źródłem czujników (albo ich podzespołów) do celów meteorologicznych i nie tylko mogą być wycofane z użycia sondy meteorologiczne (sondy tylko częściowo uszkodzone albo nie spełniające już jakichś innych wymogów).

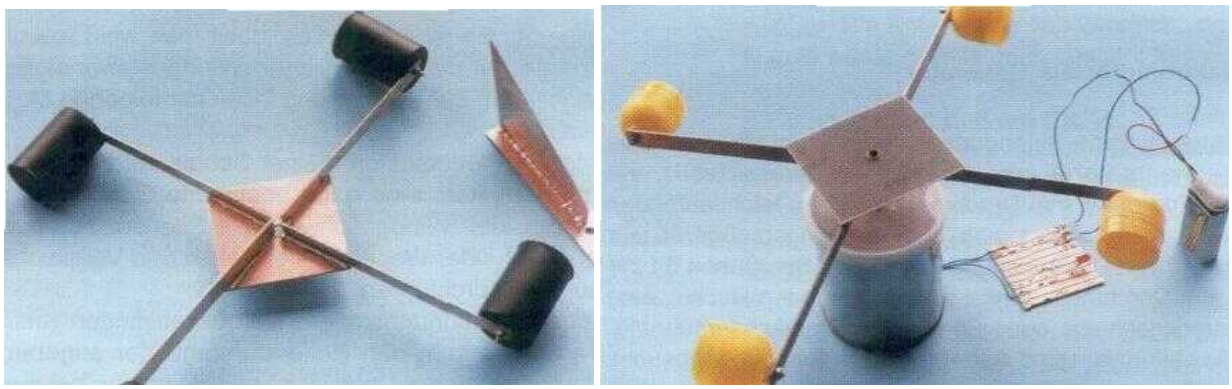
Wiatromierz

Rys. 4.1.1. Sposób wykonania wiatromierza (anemometru) z dysku CD i połówek plastikowych jajek. Dysk jest umieszczony na osi silniczka prądu stałego pracującego w tym przypadku jako generator

Silniczek prądu stałego, najlepiej bezkomutatorowy dostarcza napięcia o wartości i częstotliwości zależnych od liczby obrotów, czyli od siły wiatru.

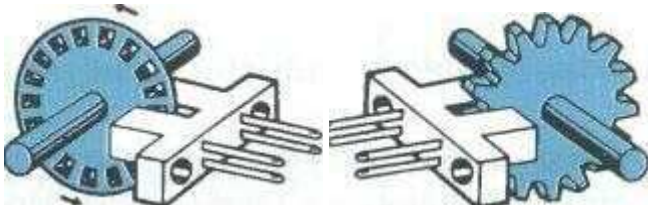


Rys. 4.1.2. Szczegół konstrukcyjny

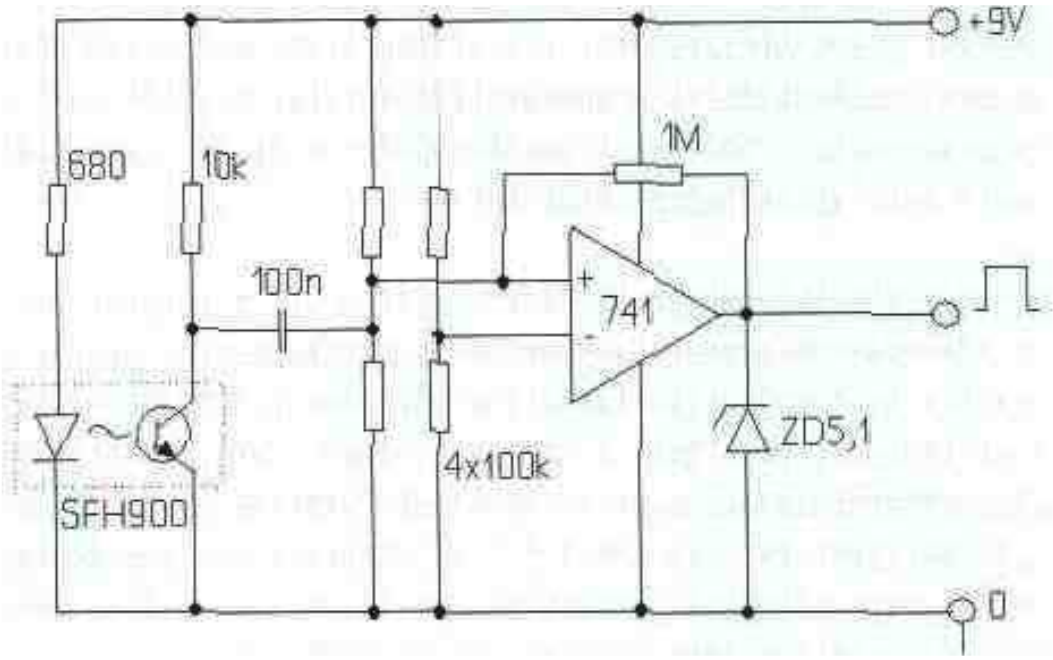


Rys. 4.1.3. Dalsze przykłady prostych konstrukcji wiatromierzy (źródło *AATiS*)

Rys. 4.1.4. Anemometr wykonany z opakowań od filmów (źródło *AATiS*)



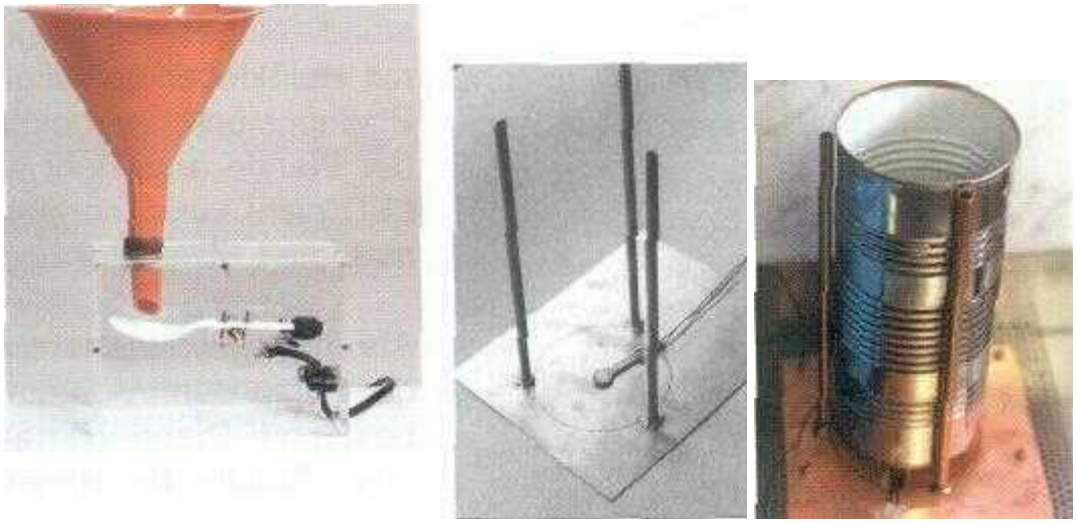
Rys. 4.1.5. Zasada pomiaru szybkości wiatru przez przerywanie strumienia światła (źródło *AATiS*)



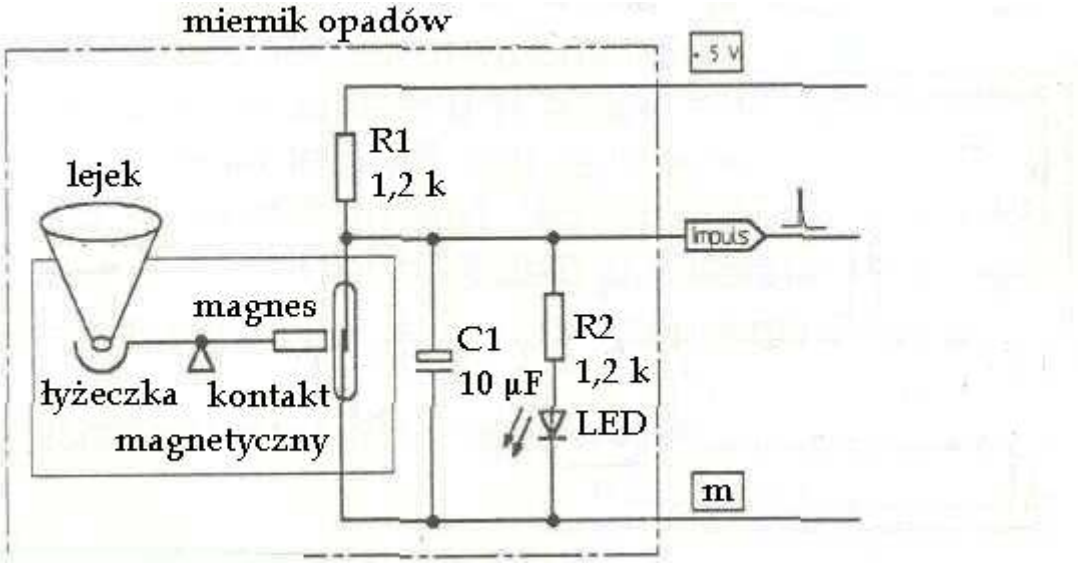
Rys. 4.1.6. Układ formowania impulsów (źródło *AATiS*)

Rys. 4.1.7. Tachometryczny sposób pomiaru szybkości wiatru (źródło *AATiS*)

Pomiar ilości opadów

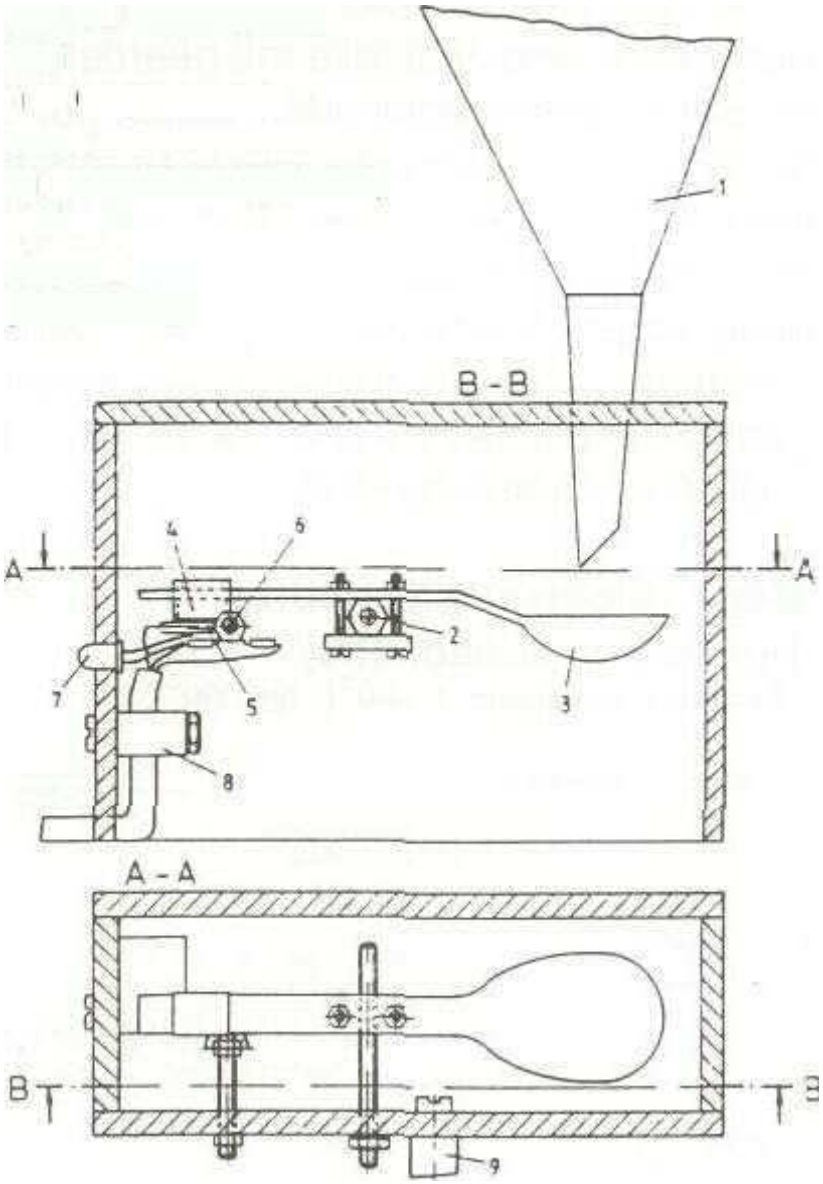


Rys. 4.2.1. Pomiar ilości opadów za pomocą ruchomej łyżeczki i pojemnika umieszczonego na czujniku ciśnienia. Po napełnieniu się łyżeczka przechyla się w dół, zwiera lub rozwiera na krótki czas kontakt dostarczający impulsów i opróżnia się. W prawej części widoczny jest pojemnik na opad umieszczony pomiędzy prętami utrzymującymi go w pożądanej pozycji na czujniku (źródło *AATiS*)



Rys. 4.2.2. Schemat ideowy miernika łyżeczkowego (źródło *AATiS*)

Elementy konstrukcyjne miernika z rys.4.2.2:
1 – lejek o średnicy 150 mm,
2 – oś obrotu z nakrętką mosiężną,
3 – łyżeczka do herbaty,
4 – magnes z przeciwwagą,
5 – kontakt magnetyczny,
6 – bolec ograniczający ruch łyżeczki,
7 – dioda świecąca,
8 – uchwyt kabla,
9 – bolec do umocowania miernika.
W celu wyskalowania miernika należy zmierzyć pipetą pojemność łyżeczki i przeliczyć to na mm słupa wody na powierzchnię lejka.

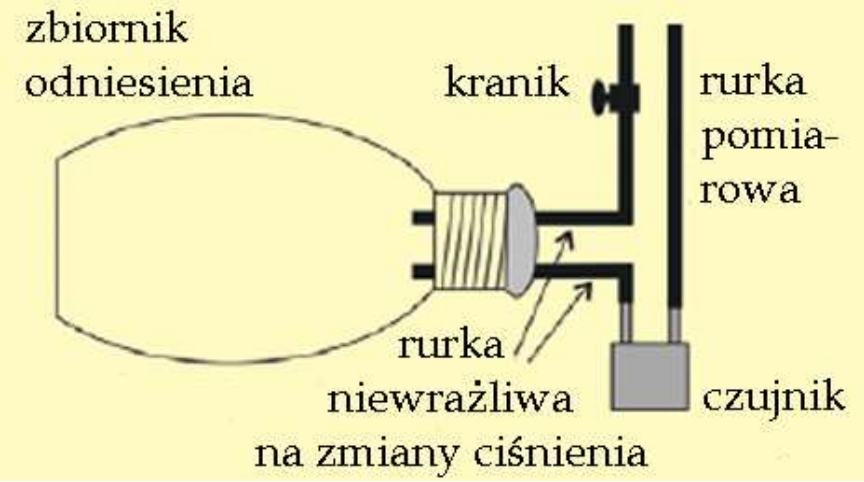


Rys. 4.2.3. Konstrukcja mechaniczna miernika (źródło *AATiS*)

Detektor infradźwięków

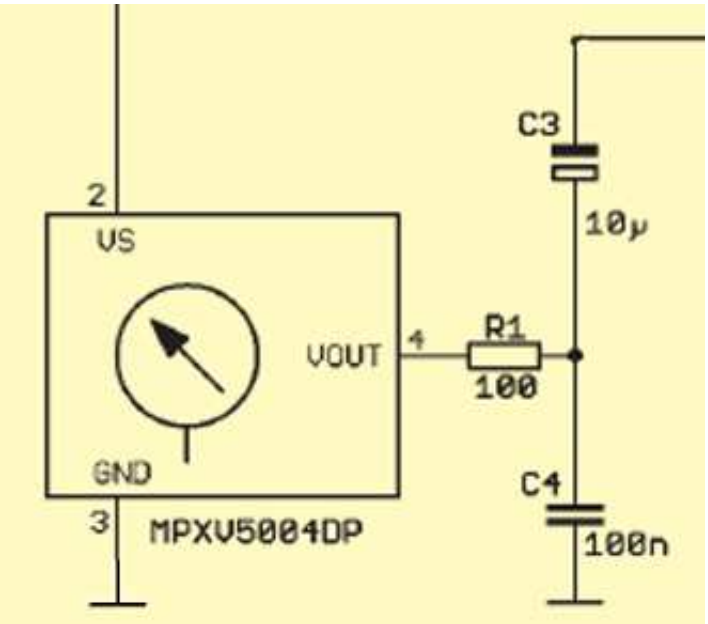
Dźwięki o częstotliwościach poniżej 16 Hz (infradźwięki) są niesłyszalne dla człowieka. Do ich wykrywania i analizy konieczne jest użycie specjalnych mikrofonów, ale można też skorzystać z czujników ciśnienia atmosferycznego. Czujniki mierzące wartość absolutną ciśnien ia atmosferycznego muszą pokrywać zakres do ponad 1000 hPa, podczas gdy infradźwięki w zależności od ich natężenia mogą powodować zmiany rzędu jednego lub kilku hPa. Oznacza to przykładowo, że przy 10-bitowym przetwarzaniu ciśnienia na wartość cyfrową różnice odpowiadają w prztybliżeniu jednemu bitowi, co jest oczywiście rozdzielczością za niską do praktycznego wykorzystania. Zamiast tego konieczne jest zastosowanie czujników różnicowych. Mierzą one różnice ciśnienia między ciśnieniem badanym i wzorcowym wywieranym przez powietrze zawarte w odpowiednio stabilnym mechanicznie i niewrażliwym na nacisk z zewnątrz zbiorniku. Jak sama nazwa wskazuje powietrze to znajduje się pod dokładnie zmierzonym ciśnieniem przyjętym jako wzorcowe. Na drugie wejście czujnika oddziałuje ciśnienie powietrza, w którym mogą się rozchodzić badane infradźwięki. Różnicowe mierniki ciśnienia charakteryzują się znaczne wyższą czułością wynoszącą przykładowo tylko kilka hPa. Pozwala to na przeprowadzanie pomiarów ze znacznie większą dokładnością (i rozdzielczością bitową).
W konstrukcji opisanej w numerze 7/2010 miesięcznika „Funkamateur” jako zbiornika odniesienia użyto szczelnie zakorkowanej butelki szklanej. Przewód do jej napełniania (co pewien czas konieczny

jest pomiar ciśnienia odniesienia i ewentualne uzupełnienie powietrza) zamykany kranikiem i przewód łączący zbiornik z czujnikiem są przeprowadzone w szczelny sposób przez korek. Muszą one być na tyle sztywne mechanicznie, aby nie ulegać odkształceniu pod wpływem ciśnienia zewnętrznego i nie powodować niepożądanych zmian ciśnienia odniesienia. Jako miernik różnicowy pracował tam MPXV5004DP zawierający scalony wzmacniacz sygnału. Maksymalny zakres mierzonych różnic ciśnień wynosi dla niego 3,92 hPa.
Zmierzone amplitudy i ewentualnie częstotliwości mogą być przekazywane radiowo do stacji odbiorczej, ale można też wykreślać czasowy przebieg wielkości mierzonej.



Rys. 4.3.1. Schemat instalacji pomiarowej

Fot. 4.3.2. Konstrukcja z wykorzystaniem butelki jako zbiornika odniesienia

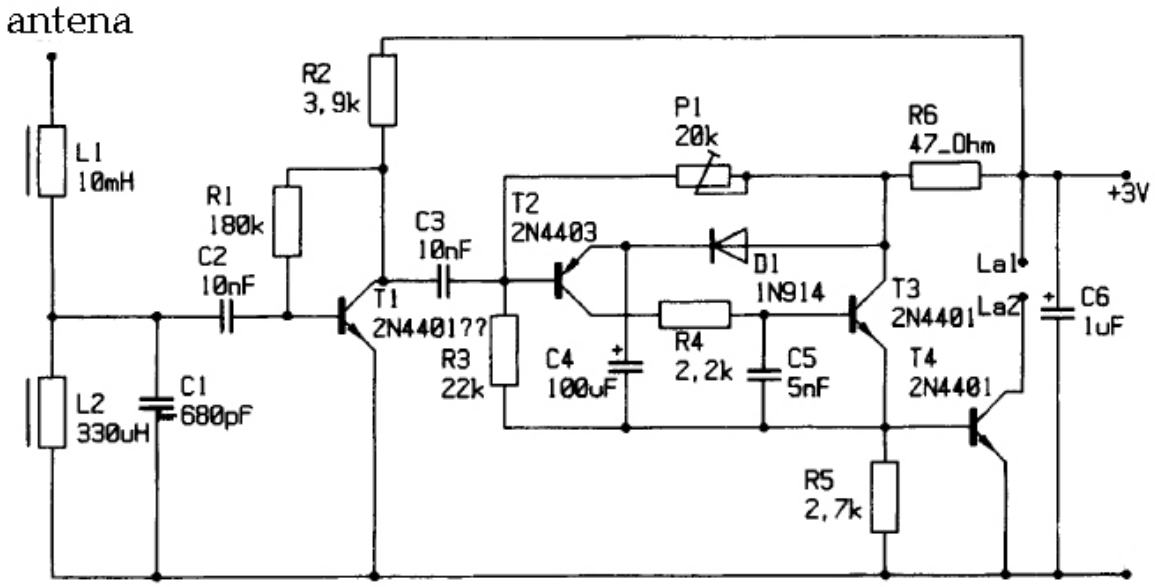


Rys. 4.3.3. Schemat podłączenia czujnika do mikrokomputera lub innego układu pomiarowego. Do wyprowadzenia 2 doprowadzone jast napięcie zasilania, a kondensator C3 jest połączony z wejściem przetwornika analogowo-cyfrowego mikrokomputera. Powinno być ono spolaryzowane napięciem stałym 2,5 V, aby możliwy był pomiar amplitud zarówno dodatnich jak i ujemnych (źródło *www.sander-electronic.de*)

Źródłami infradźwięków mogą być takie zjawiska naturalne jak trzęsienia ziemi, wybuchy wulkanów, upadki meteorytów, burze, wysokie fale morskie, wiatr, a zwłaszcza jego silne podmuchy, wiatry halne itp.
Technicznymi źródłami infradźwięków mogą być wiatraki (elektrownie wiatrowe), wybuchy, przelatujące samoloty, starty rakiet i samolotów i różne urządzenia przemysłowe.
Czujniki infradźwięków pozwalają także na zarejestrowanie przypadków otwarcia lub zamknięcia drzwi albo okien w pomieszczeniu, w którym są zainstalowane, a także wybicie szyby w oknie itp. i wszczęcie dzięki temu alarmu.

Odbiór wyładowań atmosferycznych

Odbiornik ostrzegający przed zbliżającymi się burzami jest dostrojony w przybliżeniu do częstotliwości 300 kHz, tak aby nie odbierał ani żadnych długofalowych stacji radiofonicznych ani pracujących powyżej zakresu radiowego radiolatarni lotniczych.

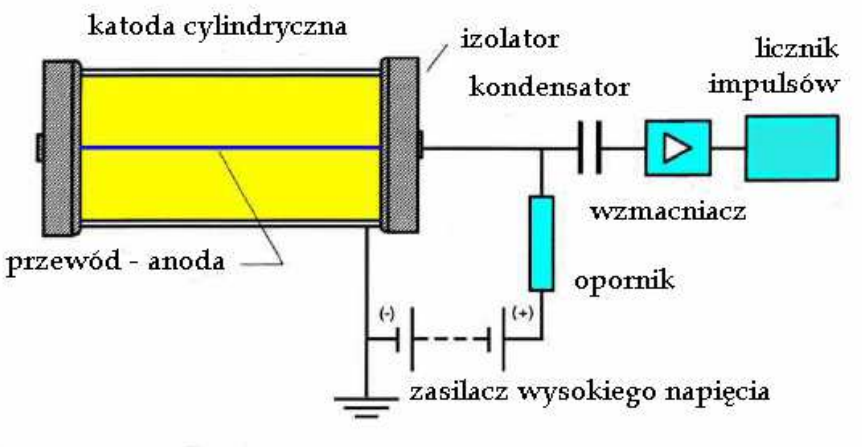


Rys. 4.4.1. Odbiornik ostrzegawczy przed burzami (źródło *AATiS*)

Antena prętowa o dlugosci 60 cm jest przedłużona elektrycznie za pomocą cewki L1 (fabrycznego dławika) o indukcyjności 10 mH. Cewka L2 i Kondensator C1 tworzą obwód rezonansowy dostrojony do częstotliwości odbioru. W obwodzie można użyć elementów o innych wartościuach po odpowiednim przeliczeniu. Podane na schemacie amerykańskie typy tranzystorów można zastąpić przez ich europejskie odpowiedniki: 2N3904 = BC546, 2N4401 = BC337, 2N4403 = BC327, i 1N914 = 1N4148 itp. Odbiornik wykrywa wyładowania atmosferyczne z odległości 100 lub nawet kilkuset km w zależności od ich siły. Do regulacji jego czułości służy potencjometr P1. Dla ułatwienia precyzyjnej regulacji można tu zastosować potencjometr 10-obrotowy. Do zacisków wyjściowych La1, La2 można podłączyć diodę świecącą, żaróweczkę 2,5 V, brzęczyk albo też można też doprowadzić impulsy wyjściowe do licznika lub wejścia mikrokomputera po ich ewentualnym uprzednim ukształtowaniu. Sprawom radiometeorologii, elektryczności atmosferycznej i wyładowań burzowych, a także układów pomiarowych i odbiorczych jest poświęcony tom 16 „Biblioteki polskiego krótkofalowca”, dlatego też tutaj nie będziemy szerzej rozwijać tych tematów.

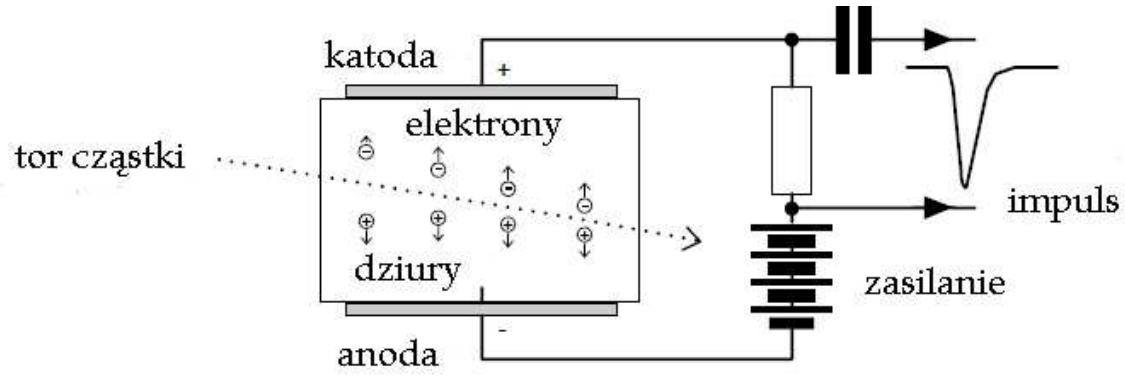
Radioaktywność

Pod pojęciem radioaktywności rozumiane jest promieniowanie cząstek powstające w wyniku rozpadu atomów oraz jonizujące promieniowanie elektromagnetryczne gamma.
Ujmując rzecz w dużym skrócie rozróżniamy promieniowanie alfa i beta. Pierwsze z nich polega na takim rozpadzie atomów pierwiastków lub ich izotopów, że oddzielają sie od niego jądra helu (zawierające dwa protony). Rozpad taki możliwy jest zasadniczo dla pierwiastków o masie atomowej przekraczającej 145. Masa atomowa jest masą wszystkich zawartych w jądrze pierwiastka protonów i neutronów. Zarówno masa atomowa protonu jak i neutronu wynosi w przybliżeniu 1. W wyniku rozpadu powstają izotopy lub kolejne pierwiastki z szeregu rozpadu o połowicznym czasie rozpadu od 10^{-7} roku do 10^{10} lat. Cząsteczki alfa są silnie pochłabniane przez materię i dlatego ich zasięg w powietrzu nie przekracza przeważnie niewielu centymetrów, a w innych materiałach jest jeszcze mniejszy.
Promieniowanie beta polega na wyrzucaniu z atomu elektronów lub pozytronów (dodatnio naładowanych eletronów będących cząstkami antymetrii). Elektrony poruszają się z prędkościami od 10^8 m/s do 3×10^9 m/s. Zasięg promieniowanna beta w powietrzu dochodzi do kilku metrów.
Promieniowanie gamma jest falą elektromagnetyczną o długościach mniejszych niż promieniowanie Rentgena. Charakteryzuje się ono większą energią i jest słabo pochłaniane przez materię.
Promieniowane neutronowe polega na wyrzucaniu z jąder neutronów w trakcie ich rozpadu. Neutrony te silnie przenikają przez materię.
Bezpośredni pomiar promieniowania radioaktywnego nie jest możliwy i dlatego wykorzystuje się do tego celu łatwo mierzalne zjawiska fizyczne z nim związane, takie jak powstawanie ładunków elektrycznych (jonizacja, zwłaszcza gazów), błyski świetlne, naświetlanie emulsji fotograficznych, powstawanie jąder luminescencji albo mętnienie szkła i proszków krystalicznych. W dalszym ciagu rozdziału przedstawimy tylko rozwiązania najłatwiejsze do realizacji dla zastosowań amatorskich i dydaktycznych.
Komora jonizacyjna posiada dwie elektrody zasilane wysokim napięciem, co oznacza, że wewnątrz niej panuje silne pole elektryczne. Przenikające do niej promieniowanie radioaktywne powoduje jonizację zawartego w niej gazu i przepływ elektrycznie naładowanych cząsteczek do elektrod – odpowiednio do ich ładunku. Elektrody mogą mieć kształt płytek lub cylidryczny.
Na zasadzie komory jonizacyjnej działają liczniki Geigera-Müllera (G-M). Komora licznika Geigera-Müllera jest wypełniona gazem szlachetnym z dodatkami metanu, propanu, etanu lub innymi. Skład gazu i jego ciśnienie są również zależne od rodzaju wykrywanych cząstek. W wyniku przepływu ładunków na oporniku obciążenia powstają impulsy elektryczne. Mogą one podlegać zliczaniu albo tylko uruchamiać sygnalizator świetlny lub dźwiękowy. W zależności od konstrukcji do zasilania licznika konieczne jest napięcie od kilkuset do nawet ponad tysiąca V. W zależności od przeznaczenia (rodzaju wykrywanych cząstek) ścianki licznika mają różną grubość, a w jednej z nich może znajdować się okienko wpustowe przysłonięte cienką folią metalową, przez którą mogą przenikać również cząstki alfa, albo cienką warstwą miki.



Rys. 4.5.1. Zasada pracy licznika Geigera-Müllera

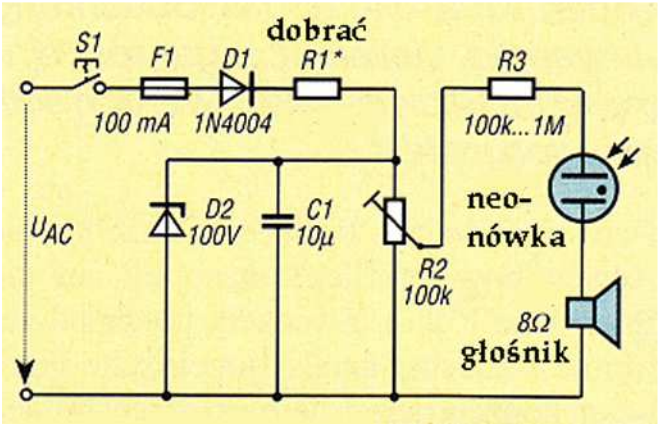
Do wykrywania promieniowania radioaktywnego stosowane są również detektory półprzewodnikowe. Są to po prostu złącza p-n wykonane w materiale o szerokiej strefie zakazanej: germanu lub krzemu z odpowiednimi domieszkami– czyli po prostu diody półprzewodnikowe spolaryzowane w kierunku zaporowym. Przenikające przez półprzewodnik cząsteczki promieniowania powodują powstanie w strefie zaporowej złącza para elektron-dziura przemieszczających się następnie w kierunku jej elektrod. W sumie jest to więc sytuacja podobna jak w przypadku komory jonizacyjnej, liczników G-M itp. W detektorach profesjonalnych stosowany jest też czysty german, a same elementy są chłodzone ciekłym azotem. W warunkach amatorskich do tego celu stosowane są fotodiody. Ponieważ gęstość ciała stałego jest znacznie większa od gęstości gazu już stosunkowo nieduża objetość wystarczy do zaabsorbowania energii cząstki. Niedogodnością detektorów półprzewodnikowych jest zależność ich parametrów od temperatury.



Rys. 4.5.2. Zasada pracy detektora półprzewodnikowego

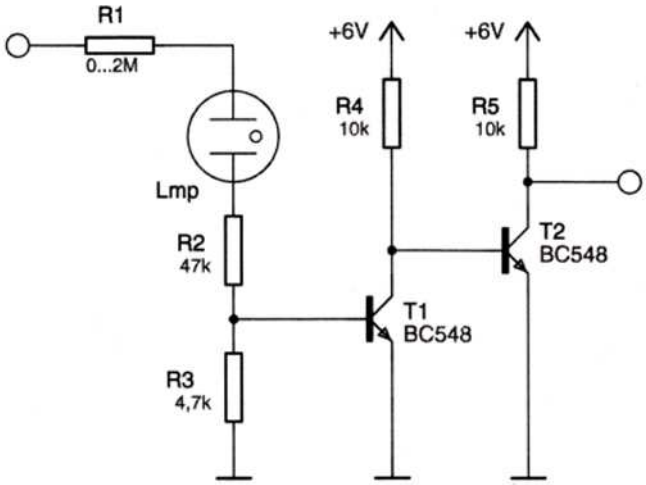
Do prób z opisanymi w daszym ciągu układami detektorów konieczne są jakieś (bezpieczne) źródła promieniowania j.np. próbki laboratoryjne materiałów radioaktywnych, zegarki z fosforyzującymi wskazówkami, stabilizatory lampowe dawniejszej produkcji (zwłaszcza dla napięć poniżej 100 V), niektóre lampy nadawcze (posiadające katody wolframowo-torowe) itp. Do ciekawszych eksperymentów z uruchomionymi układami należą badania stopnia napromieniowania róznych artykułów spożywczych lub innych codziennego użytku. Również niektóre materiały budowlane, zwłaszcza starszego pochodzenia i farby do obrazów zawierają domieszki materiałów radioaktywnych. Opisane dalej rozwiązania mają charakter eksperymentalny i nie są skalibrowane, dlatego też nawet w przypadku wskazywania przez nie zwiększonego poziomu radioaktywności nie należy wszczynać alarmu. Można jedynie poinformować się w bardziej wiarygonych źródłach o aktualnej sytuacji.

Detektor promieniowania beta i gamma na neonówce



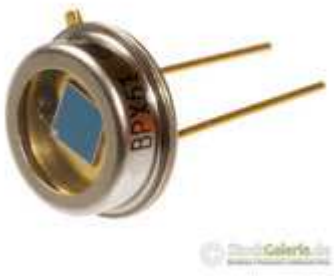
Rys. 4.6.1. Schemat ideowy

W układzie zamiast kosztownego licznika Geigera-Müllera użyto zwykłej neonówki. Jej bańka szklana pochłania jednak promieniowanie alfa, dlatego też detektor reaguje jedynie na promieniowanie beta i gamma. Układ zasilany jest napięciem zmiennym, prostowanym przez diodę D1. Do stabilizacii uzyskanego w ten sposób napięcia stałego służy dioda D2 o napięciu znamionowym 100 V. Oporność opornika R1 jest zależna od napięcia zasilania i jest obliczana ze wzoru **R1 = (Uzas – 100 V)/5 mA**.
Napięcie na neonówce jest ustawiane za pomocą potencjometru 10-obrotowego tak, aby znajdowała się na granicy zapłonu, ale jeszcze nie świeciła. Wpadające do neonówki cząsteczki promieniowania powodują jej krótkotrwałe zaświecenie (zapłon) i przepływ impulsu prądu. Napięcie na neonówce spada poniżej progu jej świecenia i dlatego też następuje jej zgaśnięcie aż do czasu przejścia następnej cząstki. Impuls prądowy jest słyszalny jako trzask w głośniku. Można go także wykorzystać do wyzwolenia przerzutnika monostabilnego dostarczającego standardowych impulsów do zliczania przez mikroprocesor lub licznik scalony albo do wyzwolenia generatora akustycznego. Układ pochodzi z nr 1/2002 „Funkamateura”.
Nie wszystkie typy neonówek nadają się jednakowo dobrze do tego celu. Elektrody niektórych z nich są pokryte materiałami powodującymi zapłon pod wpływem promieniowania ultrafioletowego, a nawet widzialnego. Pomocne w takim przypadku okazuje się światłoszczelne owinięcie neonówki cienką folią aluminiową lub jej polakierowanie.



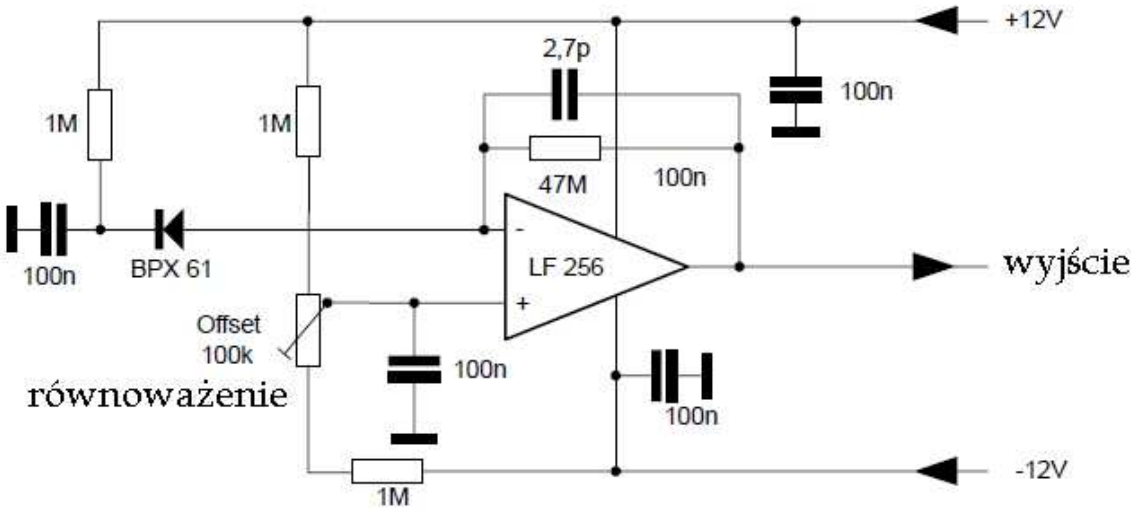
Rys. 4.6.2. Układ formowania impulsów do zliczania lub wyzwalania dalszych układów elektronicznych. Może być zasilany napięciem +5V

Detektor promieniowania na fotodiodzie



Jest to rozwiązanie niedrogie w porównaniu z licznikami G-M, przy czym szczególnie dobre rezultaty uzyskuje się przy zastosowaniu diod PIN. Są to diody, w których pomiędzy obszarami z domieszkowaniem p i n znajduje się obszar neutralny, nie domieszkowany. Uzyskany w ten sposób wzrost objętości strefy zaporowej zwiększa skuteczność wykrywania cząstek (czułość detektora).
Na ilustracji 4.7.1 przedstawiony jest schemat detektora pracującego na diodzie BPX61 (lub jej odpowiedniku). Impulsy otrzymywane na wyjściu diody są bardzo słabe i wymagaja znacznego wzmocnienia. Dla zmniejszenia niebezpieczeństwa zakłóceń elektrycznych dioda powinna być umieszczona jak najbliżej wzmacniacza. Użyto w nim scalonego wzmacniacza operacyjnego LF256. Do kompensacji wpływu prądu „ciemnego” fotodiody i napięcia nierównowagi wzmacniacza służy potencjometr montażowy 100 kΩ, który należy ustawić tak aby na wyjściu otrzymać napięcie 0 V. Warstwa zaporowa jest na tyle cienka, że detektor reaguje w pierwszym rzędzie na cząstki alfa, natomiast cząstki beta dają już tylko słabe sygnały. Konieczne jest jednak usunięcie (obcięcie) szklanego okienka osłaniającego strukturę diody, aby nie absorbowało ono cząstek alfa. Diodę

i cały układ należy umieścić w obudowie światłoszczelnej, na przykład w puszce metalowej. Dla zmniejszenia pochłaniania cząstek alfa przez powietrze można je z niej częściowo wypompować.

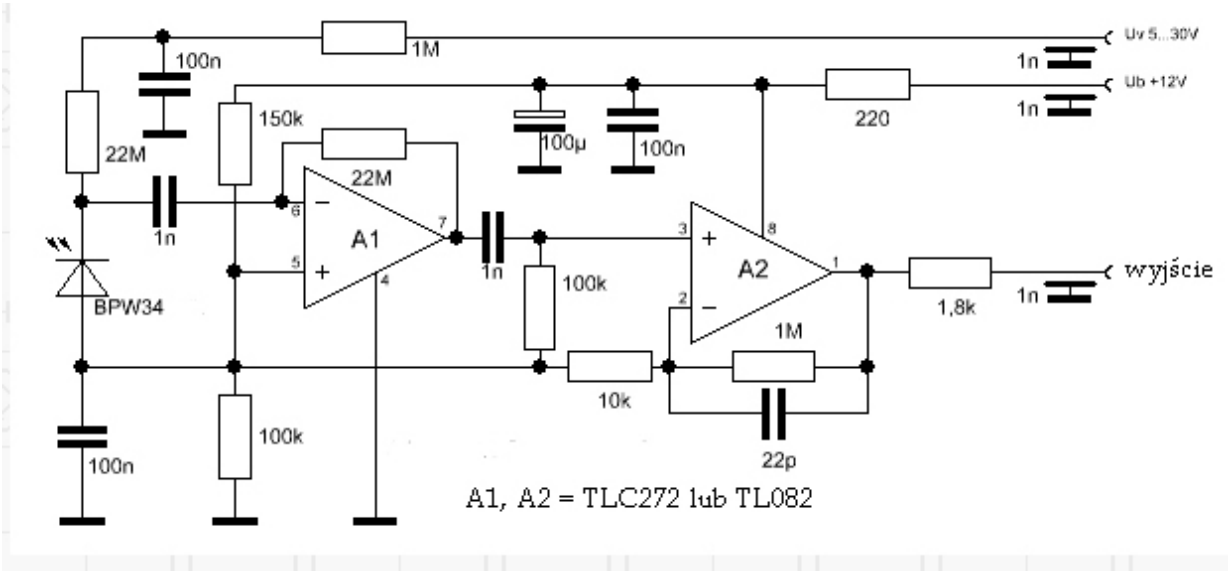


Rys. 4.7.1. Detektor na fotodiodzie BPX61

W porównaniu z diodami BPW34 i podobnymi BPX61 wykazuje większą czułość. Dioda bez usuniętego okienka nadaje się do detekcji promieniowania gamma.

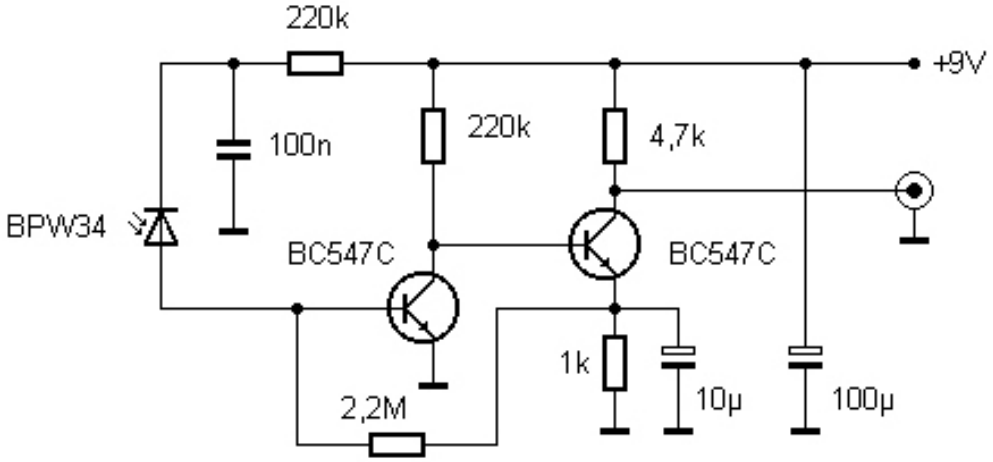
Detektor promieniowania gamma na fotodiodach PIN

W ukladzie detektora zastosowano diodę PIN typu BPW34, BPW41N, BPW21 (po usunięciu szklanego lub plastikowego okienka) lub podobną.Całość układu ze względu na duże wzmocnienie wymaga dobrego ekranowania. W obudowie konieczne jest wykonanie otworka przepuszczającego promieniowanie do fotodiody. Otworek można przysłonić taśmą izolacyjną aby do fotodiody nie docierało światło. Jeżeli ekranowanie okaże się niewystarczające i do układu będzie przenikać przydźwięk sieciowy można go dodatkowo umieścić w puszce metalowej. Zastosowanie zamiast pojedyńczej diody kilku diod połączonych równolegle daje wzrost amplitudy impulsów.
Przeprowadzone próby wykazały, że fotodiody PIN można stosować także do detekcji promieniowania Rentgena.

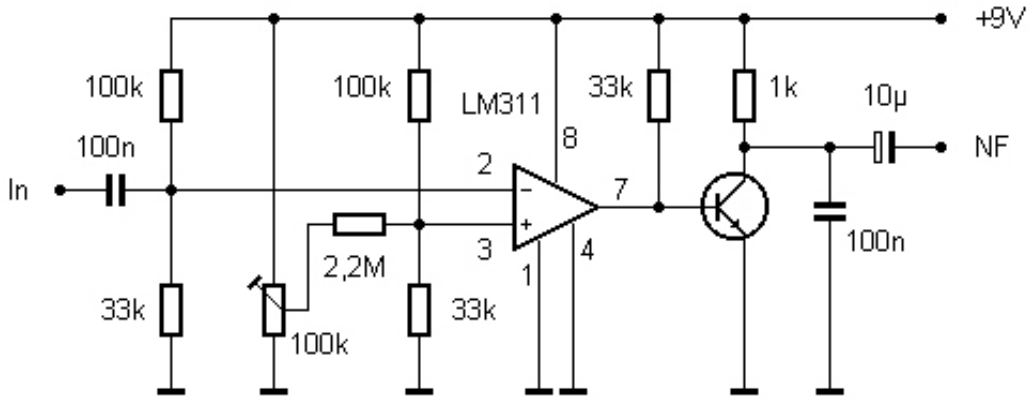


Rys. 4.8.1. Schemat ideowy (*www.elektronik-labor.de*)

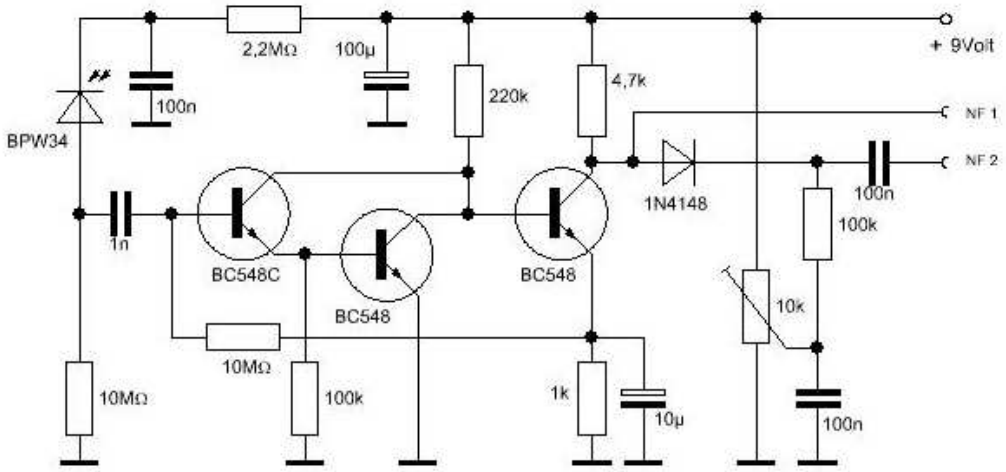
Rys. 4.8.2. Układ w obudowie ekranującej (*www.elektronik-labor.de*)



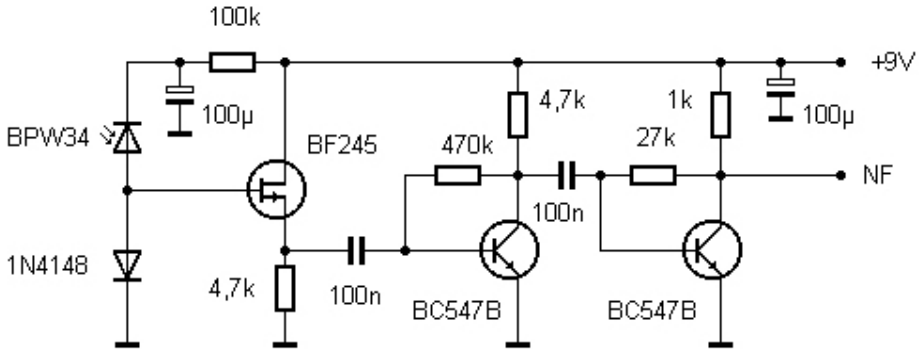
Rys. 4.8.3. Detektor cząstek alfa na diodzie BPW34 z dwutranzystorowym wzmacniaczem (*www.elektronik-labor.de*)



Rys. 4.8.4. Układ komparatora i przedłużania impulsów. Impulsy stają się dzięki temu słyszalne, ale mogą być też zliczane elektronicznie (*www.elektronik-labor.de*)

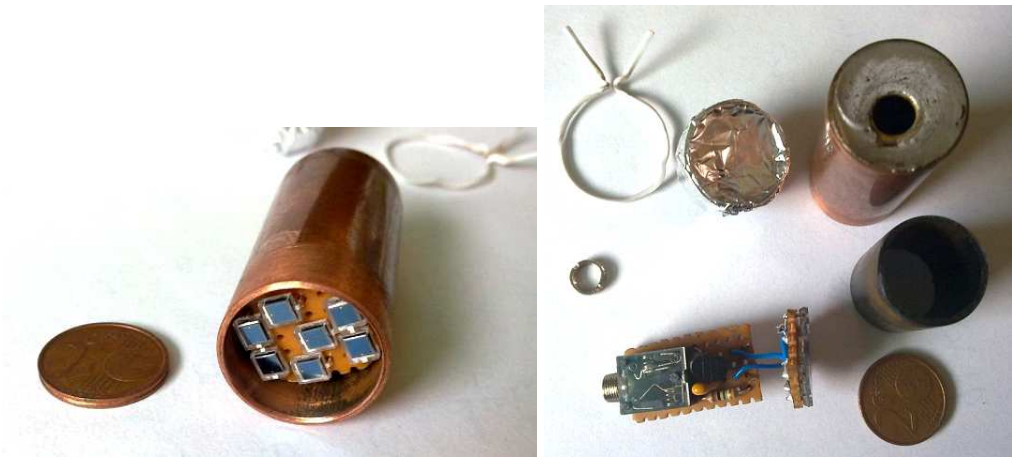


Rys. 4.8.5. Układ detektora z dwustopniowym wzmacniaczem, wtórnikiem emiterowym i wyjściami m.cz. i dla licznika impulsów. Tranzystory pracujące w pierwszych stopniach powinny wnosić jak najmniej szumów (*www.elektronik-labor.de*)

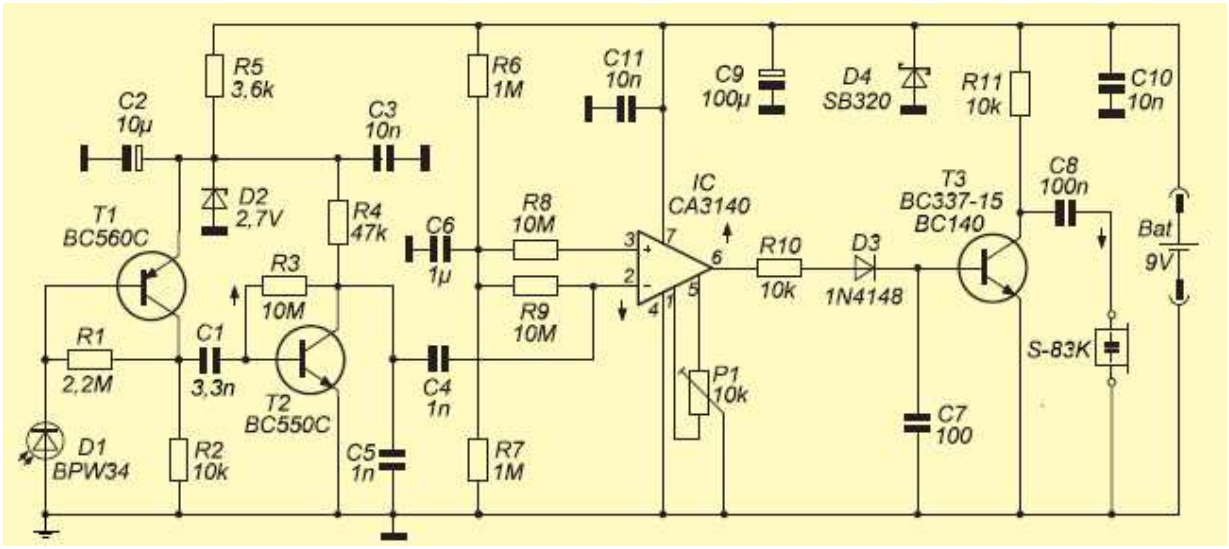


Rys. 4.8.6. Detektor promieniowania gamma na BPW34 (*www.elektronik-labor.de*)

Dioda 1N4148 służy do ustalenia punktu pracy tranzystora. Dioda BPW34 powinna być chroniona przed swiatlem. Całość najlepiej jest umieścić w jakiejść puszce metalowej np. po biszkoptach itp. W odróżnieniu od detektorów cząstek alfa nie jest tu konieczne usunięcie okienka szklanego.



Fot. 4.8.7a–b. Detektor kilkudiodowy wbudowany do rurki miedzianej przysłanianej z przodu możliwie jak najcieńszą folią aluminiową, np. 3/100 mm. Diody są połączone równolegle (*www.elektronik-labor.de*)



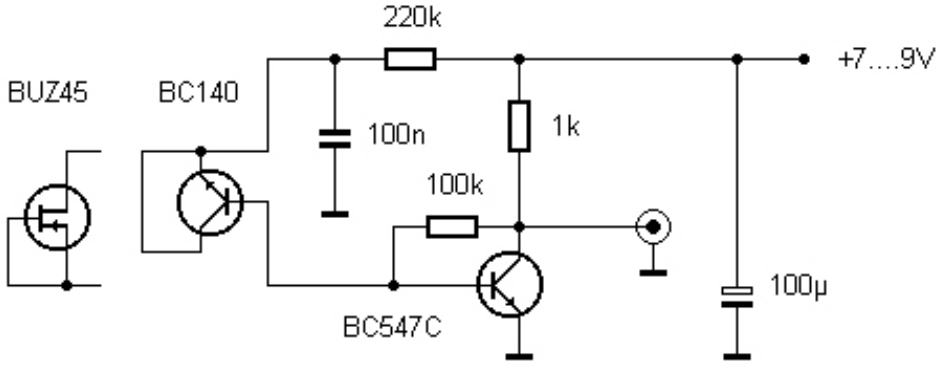
Rys. 4.8.8 Detektor promieniowania gamma z sygnalizacją dźwiękową na słuchawkę

Płytkę z układem najlepiej umieścić w metalowej puszce np. od pasty do butów, którą należy uszczelnić tak aby nie dopuścić do niej wogóle światła. Powinna ona być umieszczona w odległości kilku (np. 3) mm od denka puszki. Gniazdko do wysokoomowej słuchawki (S-83K lub podobnej) należy umieścić tak aby było izolowane elektrycznie od puszki-obudowy. Tranzystory T1 i T2 powinny mieć jak najniższy poziom szunów własnych, powinny zostać pod tym kątem wyselekcjonowane z większej liczby. C1 jest kondensatorem foliowym. Dioda D1 powinna być umieszczona równolegle do ścianki puszki, ale nie może jej dotykać. Przewody zasilające prowadzące do baterii są wypuszczone przez otwory w puszce i do nich przyklejone. Czułość układu jest regulowana za pomocą potencjometru montażowego P1. Rozwiązanie zostało opisane w nr 6/2013 miesięcznika „Funkamateur”.

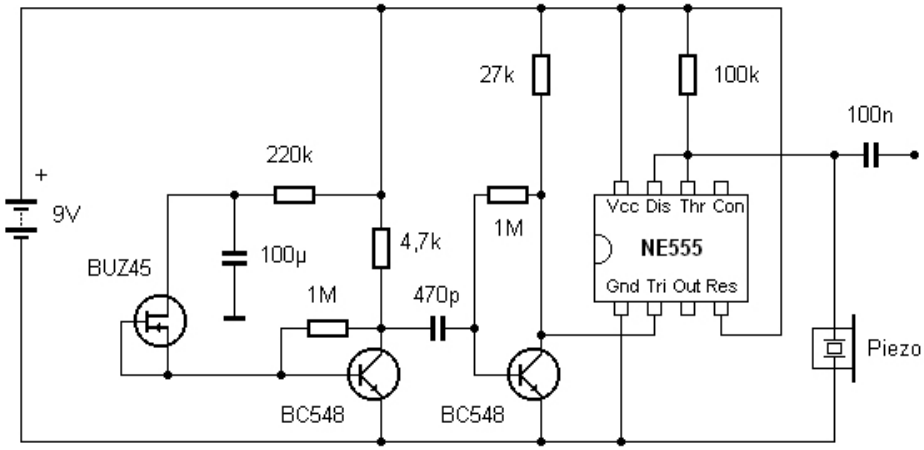
Detektory promieniowania na tranzystorach

Zamiast fotodiod jako detektorów można użyć różnych typów tranzystorów po odpiłowaniu górnej części ich obudowy i odsłonięciu struktury półprzewodnikowej. Pozwala to m.in. na detekcję promieniowania alfa. Prowadzone były próby z tranzystorami w obudowach TO-3, BUZ45, 2N3055 lub podobnymi w obudowach TO-5, BC140 lub podobnymi (*www.elektronik-labor.de*). Struktura półprzewodnikowa nie może być przykryta żadną warstwą ochronną żelu ani innej substancji. W razie jej obecności należy ją usunąć.

Fot. 4.9.1. Tranzystor BUZ45 z otwartą obudową. Struktura półprzewodnikowa pokryta warstwą ochronną



Rys. 4.9.2. Detektor tranzystorowy ze wzmacniaczem jednostopniowym. Złącza B-E i B-C tranzystora BC140 są połączone równolegle (*www.elektronik-labor.de*)



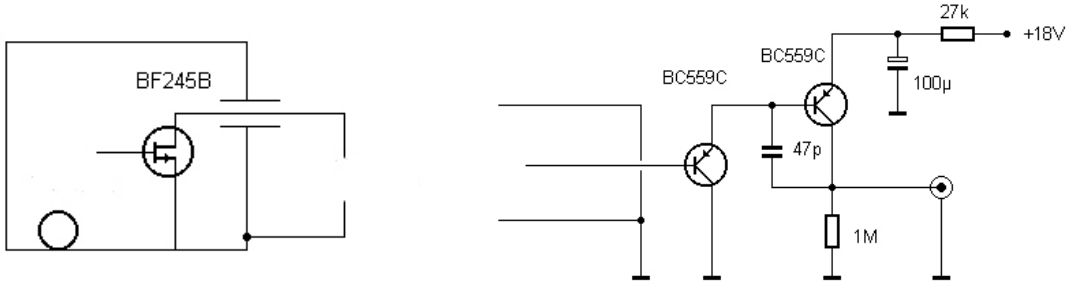
Rys. 4.9.3. Detektor tranzystorowy z sygnalizacją dźwiękową. NE555 przedłuża krótkie impulsy z detektora dzięki czemu stają się słyszalne (*www.elektronik-labor.de*)

Amatorska komora jonizacyjna do detekcji cząstek alfa

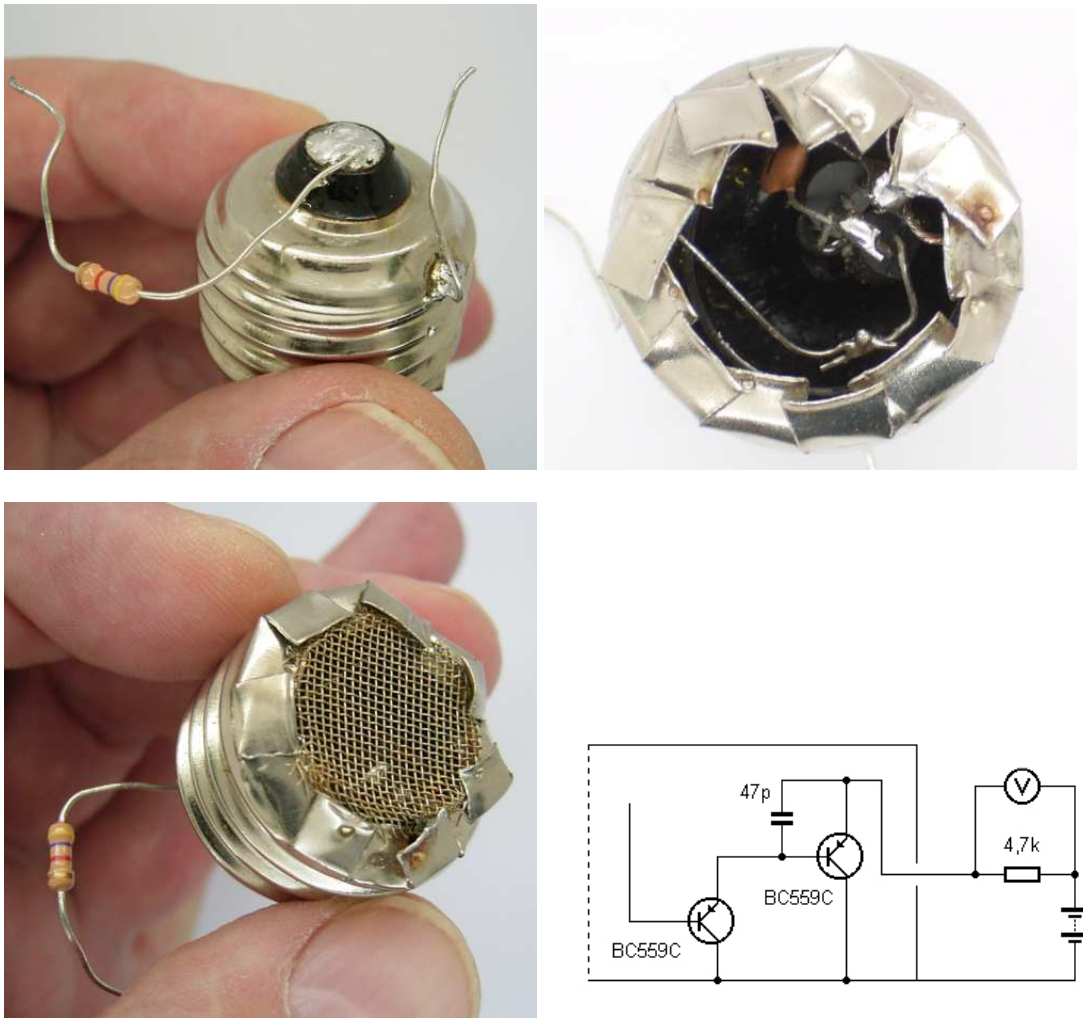
Fot. 4.10.1. Konstrukcja otwartej komory detektora (*www.elektronik-labor.de*)

Przedstawiona na fotografii konstrukcja wykorzystuje przejściówkę UHF-BNC. Do ochrony przewodu stanowiącego elektrodę środkową komora służy widoczna na zdjęciu siatka metalowa. Przenikające do

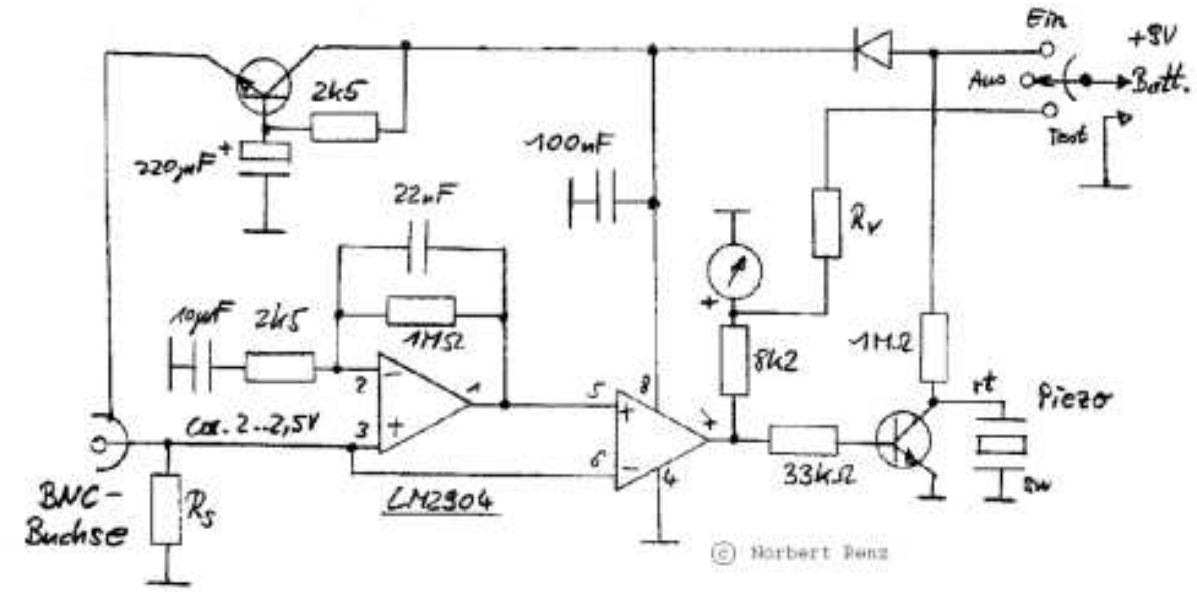
komory cząstki powodują jonizację gazu, przy czym elektrony jako bardziej ruchliwe szybciej docierają do obudowy, a we wnętrzu przez stosunkowo dłuższy od tego czas utrzymują się dodatnie jony gazu. Elektroda środkowa może być połączona z bramką tranzystora polowego MOSFET lub JFET albo z bazą tranzystora złączowego.



Rys. 4.10.2. Układ elektryczny głowicy pomiarowej (*www.elektronik-labor.de*)

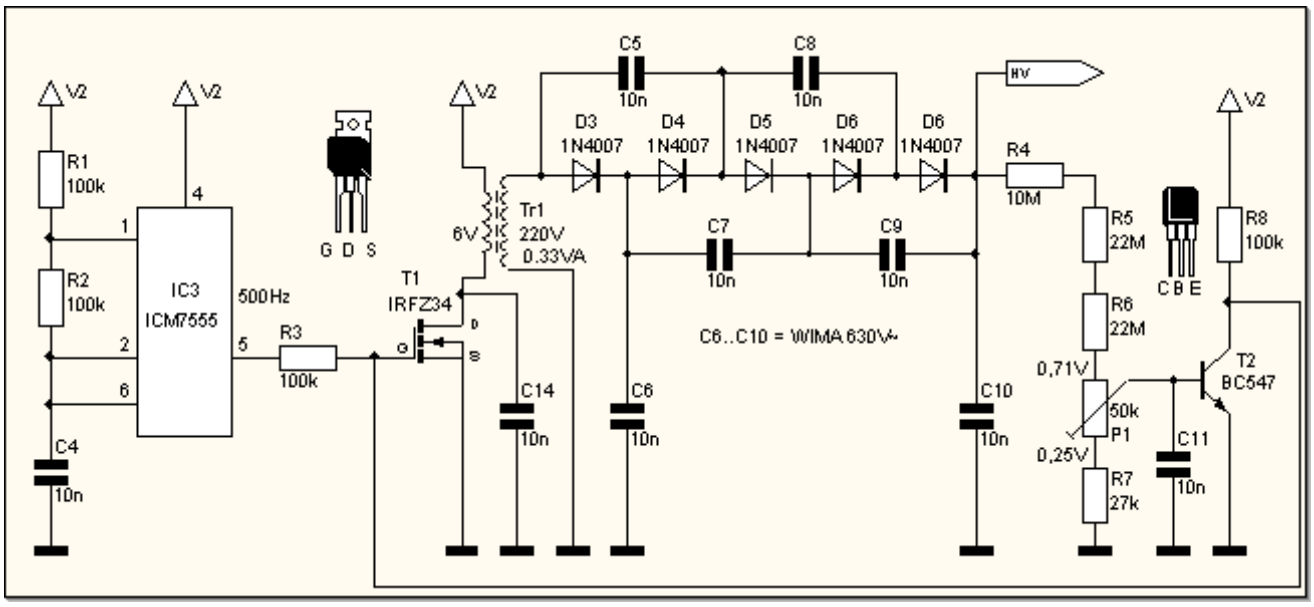


Ilustr. 4.10.3a-d. Konstrukcja komory z gwintu (podstawki) żarówki i schemat układu elektrycznego (*www.elektronik-labor.de*)



Rys. 4.10.4. Układ pomiarowy z sygnalizacją dźwiękową dla głowicy nr 1 (*www.elektronik-labor.de*)

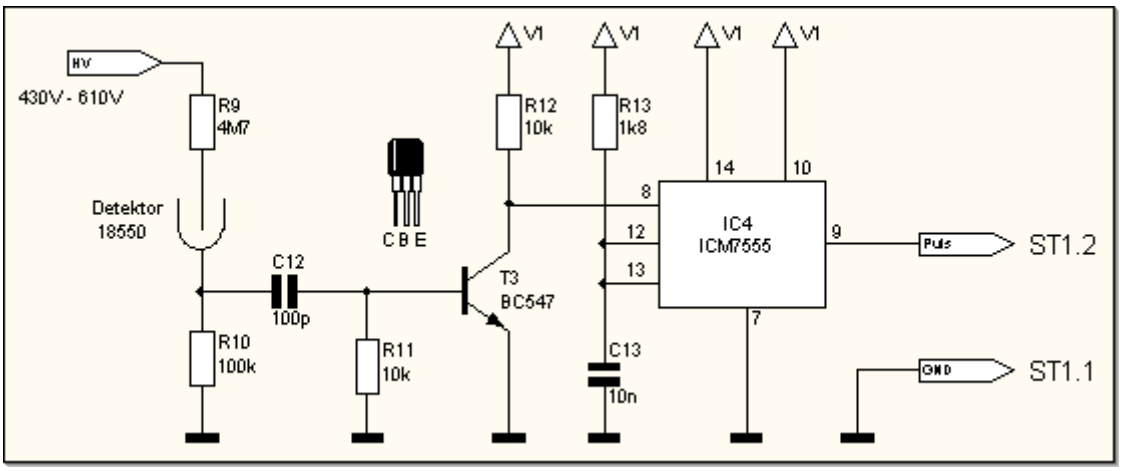
Detektor promieniowania gamma z licznikiem Geigera-Müllera



Rys. 4.11.1. Układ zasilacza wysokiego napięcia

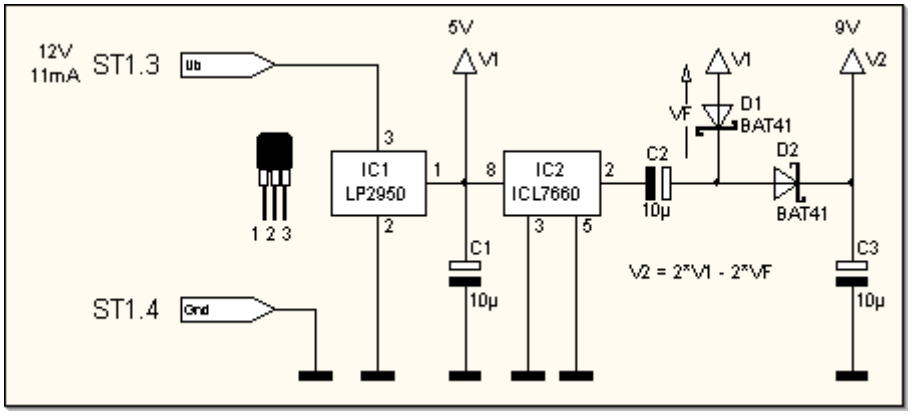
W rozwiązaniu tym zastosowano licznik typu 18550 firmy Philips, którego odpowiednikami są ZP1320 (ZP1400, ZP1401) firmy Centronics, 713 firmy LND, MX164 firmy Mullard lub N117-1/C1320 firmy „TGM Detectors". Dostosowanie układu do innych typów liczników może w pierwszym rzędzie wymagać zmiany zasilającego je wysokiego napięcia. Licznik ma kształt rurki wykonanej ze stali nierdzewnej i jest wypełniony mieszaniną neonu i halogenu. Dopuszczalny zakres temperatur otoczenia w trakcie pracy wynosi -40 – +75 ºC, a zalecane napięcie zasilania 500 V (450 – 650 V).
Układ zasilacza wysokiego napięcia składa się z generatora 500 Hz pracującego na obwodzie scalonym ICM755. Generator ten steruje tranzystor IRFZ34 dostarczający napięcia prostokątnego do uzwojenia pierwotnego transformatora 6/220 V (może być to przykładowo transformator dzwonkowy albo od zasilacza niskiego napięcia). Po powieleniu napięcia z uzwojenia wtórnego za pomocą powielacza na diodach D3 – D6 otrzymywane jest w przybliżeniu napięcie stałe 630 V. Potencjometr montażowy P1 w obwodzie bazy tranzystora T2 służy do regulacji napięcia wyjściowego zasilacza. Po przekroczeniu ustawionego napięcia wyjściowego tranzystor T2 zaczyna przewodzić i blokuje T1 do czasu gdy

napięcie wyjściowe nie obniży się. W zależności od pożądanego zakresu napięć wyjściowych konieczne może być dobranie opornika R4.



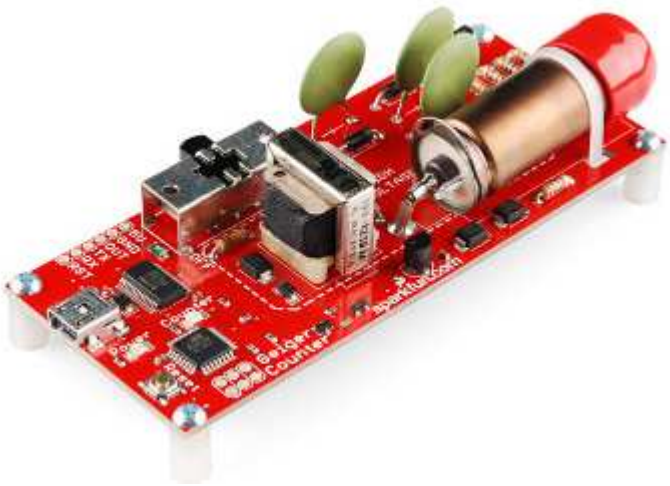
Rys. 4.11.2. Układ licznika i formowania impulsów

Opornik R9 z licznikiem Geigera-Müllera i opornikiem R1 tworzą dzielnik napięcia. W przypadku wykrycia kwantu promieniowania gamma na jego wyjściu powstają impulsy o amplitudzie około 10 V. Poprzez tranzystor T2 wyzwalająone przerzutnik monostabilny na obwodzie ICM7555, który generuje impulsy o szerokości 2 µs. Są one następnie podawane na układ zliczający (może być to też wejście przerwań mikrokomputera).



Rys. 4.11.3. Zasilacz 5/9 V. Diody D1 i D2 są diodami Schottkiego

Fot. 4.11.3. Licznik Geigera-Müllera typu LND712 dla promieniowania alfa, beta i gamma jest wypełniony mieszaniną neonu i halogenu pod niskim ciśnieniem. Wysokie napięcie leży w zakresie 325 – 500 V



Fot. 4.1.4. Fabryczny moduł z licznikiem LND-712

Standardy transmisji

APRS

System APRS polega na nadawaniu krótkich komunikatów zawierających współrzedne geograficzne stacji i ewentualne dodatkowe dane telemetryczne albo meteorologiczne przy użyciu nienumerowanych pakietów UI protokółu AX.25. Pakiety te są przeznaczone dla wszystkich odbierających je stacji, które nie muszą nawiązywać połączenia z nadawcą, ale nie mogą też kwitować prawidłowo odebranych danych ani żądać ich powtórzenia w przypadku wystąpienia przekłamań w transmisji. Pakiety te mogą być retransmitowane przez cyfrowe przemienniki. W miarę możliwości są one również przekazywane do bramek radiowo-internetowych albo radiowo-hamnetowych.
Sieć przemienników cyfrowych APRS pracuje zasadniczo na jednej częstotliwości, a jedynie w rejonach o dużym natężeniu ruchu używane są dodatkowe kanały, przeważnie w paśmie 70 cm. W Europie główną częstotliwością jest 144,800 MHz, a do częstotliwości pomocniczych należą 432,500 i 433,800 MHz. W pasmach UKF stosowana jest przepływność 1200 bit/s i kluczowanie AFSK. Stacje użytkowników indywidualnych mogą być wyposażone w modemy TNC wbudowane do radiostacji UKF niektórych typów (TH-D72E, TH-D74E, TM-D710E), w modemy niezależne j.np. TNC-2, PK-232, SP-232, SCS-Tracker, PTC-3, DR-7400 itp. lub w modemy pracujące na mikrokomputerach i posiadające wbudowane odbiorniki GPS oraz wejścia dla sygnałów pomiarowych. Te ostatnie zgłaszają do sieci automatycznie położenie stacji wraz z ewentualnymo dodatkowymi danymi. Modemy takie bardzo często noszą fabryczne nazwy *Track* lub *Tracker* w rozmaitych wariantach i z różnymi dodatkami. Stacje stałe nie muszą być nawet wyposażone w odbiornik GPS, wystarczy wprowadzenie pozycji do konfiguracji oprogramowania. Najprostsza stacja APRS składa się z modemu TNC i radiostacji na pasmo 2 m. Może być to przykładowo fabryczna radiostacja przenośna albo radiostacja własnej konstrukcji oparta na modułach DRA1818 lub podobna.
Komunikaty odebrane radiowo mogą być wyświetlane na tle map przez takie programy jak UI-View, WinAPRS i podobne, a komunikaty przekazane do serwerów internetowych APRS-IS – przykładowo pod adresem *aprs.fi*. Szczegółowe informacje na temat systemu APRS zawiera tom ósmy niniejszej serii „APRS i DPRS”. Adresem docelowym pakietów jest często APRS, GPS, WX (dla stacji meteorologicznych) lub TLC, TLH – dla stacji telemetrycznych.
W definicji standardu APRS przewidziano szereg formatów pakietów mogących oprócz współrzędnych geograficznych zawierać dowolne teksty, dane meteorologiczne albo telemetryczne. Czytelnikom zainteresowanym szczegółami można polecić poz. [111]. Do uruchomienia prostej radiolatarni APRS wystarczy jednak skorzystanie z jednego z podanych poniżej przykładów (dla poprawy czytelności są one podane w cudzysłowach, które należy pominąć wprowadzając komunikaty do programu terminalowego lub TNC):

1) "=5017.75N/02116.10E-dowolny tekst o długości do 43 znaków alfanumerycznych" – komunikat radiolatarni zawierający dowolny tekst uzupełniający o długości do 43 znaków alfanumerycznych rozpoczyna się od znaku równości i zwiera następnie szerokość geograficzną stacji w formacie stopnie, minuty i sekundy przeliczone na ułamek dziesiętny np. dla szerokości 50º17'45" należy sekundy przeliczyć na ułamek dziesiętny (45/60 = 0.75) i dodać do liczby złożonej ze stopni i minut – N oznacza tu szerokość geograficzną północną. W analogiczny sposób należy przeliczyć długość geograficzną – w przykładzie jest to 21º16'06" długości wschodniej. Znaki "/" (rozdzielający współrzędne geograficzne) i "-" (przed tekstem) służą do wyboru symbolu wyświetlanego na mapach APRS. Dla stacji stałej pracującej w zakresie UKF (w paśmie 2 m jest to przeważnie częstotliwość 144,800 MHz) jest to podana w przykładzie kombinacja "/" i "-", dla stacji stałej pracującej w zakresie KF będzie to kombinacja "\" i "-", a dla stacji meteorologicznej kombinacja „/” i „_” lub „/” i „W”. Kombinacja znaków „/” i „E” daje symbol oka, co można by też rozumieć jako dowolnego rodzaju stację obserwacyjną (lub pomiarową). Najczęściej występujące symbole i ich kody podano w tomie 8 „Biblioteki”. Zakończeniem komunikatu mogą być np. dowolne dane telemetryczne lub

meteorologiczne podane otwartym tekstem. Własne współrzędne geograficzne można odczytać z mapy bądź z pożyczonego odbiornika GPS. Odbiornik ten nie jest jednak stale potrzebny.

2) "!5017.75N/02116.10E-" – jest to komunikat analogiczny do poprzedniego ale nie zawierający dodatkowego tekstu i z tego powodu poprzedzony wykrzyknikiem.

3) "@271510z5017.75N/02116.10E-dowolny tekst o długości do 43 znaków alfanumerycznych" – komunikat podobny do pierwszego ale zawierający na początku datę w formacie: dzień bieżącego miesiąca, godzina i minuty czasu UTC. Dla czasu lokalnego należy zamiast litery z umieścić ukośną kreskę "/". Zalecane jest jednak podawanie czasu UTC.

4) "/271510z5017.75N/02116.10E-" – komunikat analogiczny do 3) ale bez dodatkowego tekstu.

Tekst nadawany na zakończenie komunikatu może także zawierać dane meteorologiczne w jednym ze standardowych formatów. Dokument [5] zawiera przykłady formatów stosowanych m.in. przez automatyczne stacje meteorologiczne dostępne na rynkach USA i krajów zachodnioeuropejskich. Ceny takich stacji są jednak dość wysokie dlatego też jako przykład wybrano tylko jeden z ustalonych formatów – stosunkowo najłatwiejszy do utworzenia ręcznie i do odczytania. W polu tekstowym komunikatów 1) lub 3) należy wprowadzić dane meteorologiczne w podanej poniżej postacji:

cxxx – kierunek wiatru w stopniach, litera "c" i 3 cyfry; w przypadku braku danych mogą to być trzy kropki lub trzy znaki odstępu.

sxxx – szybkość wiatru w km/h również trzycyfrowo.

gxxx – szybkość wiatru w porywach w ciągu ostatnich 5 minut, trzycyfrowo.

txxx – temperatura w ºC z ewentualnym znakiem minus.

rxxx – opady w ciągu ostaniej godziny w mm.

pxxx – suma opadów w ciągu ostatnich 24 godzin.

Pxxx – suma opadów od północy bieżącego dnia.

hxx – wilgotność względna w %.

bxxxxx – ciśnienie atmosferyczne w dziesiątych częściach hektopaskala, a więc np dla 1025 hPa – 10250.

Pierwsze cztery pola (c, s, g, t) są obowiązkowe i muszą być wprowadzone w podanej kolejności, a następne pola można dowolnie opuszczać lub zmieniać ich kolejność. W miejsce brakujących danych należy wprowadzać kropki lub znaki odstępu. Symbol stacji meteorologicznej wywołuje podkreślnik.

Przykład komunikatu z dn. 8 bieżącego miesiąca, godz. 12.00 UTC – ciśnienie 1025 hPa, temperatura -20 ºC, pozostałych danych brak:

"=5017.75N/02116.10E_c...s...g...t-20b10250"

Utworzone w ten sposób teksty komunikatów można następnie wprowadzić do dowolnego programu terminalowego Packet Radio lub do modemu TNC (np. SP-232/PK-232).

Specyfikacja APRS przewiduje poza tym specjalne formaty pakietów telemetrycznych nie zawierających ani współrzędnych ani czasu.

Komunikat taki może wyglądać następująco:

OE1KDA-2>APRS

T#011,155,188,000,000,000,00000000

gdzie T oznacza komunikat telemetryczny, #011 kolejny numer pakietu, tutaj jest to numer 11, po nim następuje 5 wartości pomiarowych, a na pozycji ostatniej (zapisane czerwonym kolorem) podawane są poziomy logiczne 8 bitów informujących o stanie pracy urządzenia lub stosowanych do zdalnego sterowania jeżeli jest to potrzebne. Przekazywane są 8-bitowe wartości mierzone, czyli liczby w zakresie 0 – 255. Pierwsza wartość (155) oznacza temperaturę i wymaga ona oczywiście odpowiedniego przeliczenia. Dla scalonego termometru LM335 włączonego w szereg z opornikiem 6,8 kΩ stosowany jest wzór T = 1,9608 X – 273 [ºC], gdzie X jest wartością transmitowaną (tutaj równą 155).

Po przeliczeniu odpowiada to 30,955 ºC czyli w przybliżeniu 31 ºC.

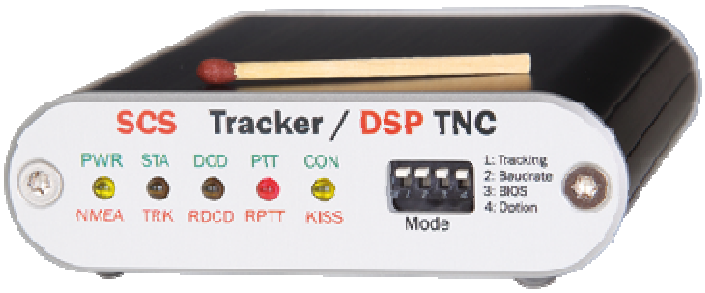
Wartość 188 na drugiej pozycji odpowiada w tym przykładzie napięciu zasilania. Jest ono mierzone za pomocą dzielnika 27 kΩ/10 kΩ. Wartość nadawana musi być więc podzielona przez 13,88 co oznacza, że napięcie zasilania wynosi w tym przypadku U = 188/13,78 = 13,64 V. Oczywiście w zależności od zakresu mierzonych napięć należy dobrać odpowiedni dzielnik oporowy i dostosować do niego obliczenia. Na pozostałych trzech pozycjach mogą być nadawane dowolne wielkości mierzone przy użyciu dowolnych czujników. Mogą być to przykładowo drugi termometr do pomiaru temperatury na zewnątrz albo w innych miejscach (np. KTY81), miernik oświetlenia, czujnik wilgotności, barometr itp.

Zmierzona wartość występuje jako dzielnik we wzorze U = 255/X, po czym w zależności od charakterystyki czujnika konieczne mogą być dalsze przeliczenia na odpowiednią wielkość fizyczną.
W przypadku ogólnym do przeliczenia stosowany jest wzór
$U = aX^2 + bX + c$, gdzie X jest wartością transmitowaną. Jeżeli mierzone wielkości, wzory i występujące w nich współczynniki są znane odbiorcom transmisja danych i ich interpretacja nie stanowią żadnego problemu. Dla elementów pomiarowych o charakterystyce liniowej współczynnik **a** przyjmuje wartość 0.
W przypadku gdy informacje te mogą być nieznane przynajmniej części odbiorców trzeba posłużyć się dodatkowymi komunikatami zawierającymi informacje o mierzonych wielkościach, jednostkach i współczynnikach występujących w ogólnym wzorze:
PARM. Temperatura,Napiecie,Wilgotnosc,Oswietlenie,Cisnienie_atm.
UNIT. stp. C,V,procent,lux,hPa
EQNS. 0,1.961,-273,0,0,0.72,0,0,0.39,0,0,0.393,0,0,1,0,0,0.706,888
Komunikat pierwszy informuje o mierzonych wielkościach fizycznych, w kolejności ich nadawania, w drugim podane są w tej samej kolejności używane jednostki, a w trzecim kolejno (zawsze) wszystkie trzy współczynniki równań, ogólnie komunikat ten ma format EQNS.A1,B1,C1,A2,B2,C2,A3,B3C3 itd.

Komunikaty o formacie APRS można także nadawać jako teksty przy użyciu dowolnych modemów i radiostacji SSB albo FM w zależności od zakresu częstotliwości lub korzystając z transmisji danych w systemach cyfrowego głosu D-STAR, C4FM, a także w formacie D-PRS.
Jedną z takich możliwości jest transmisja w opisanym poniżej systemie „LoRa". Krótkofalowcy widzą w nim następcę klasycznego APRS, ale oczywiście komunikaty „LoRa" mogą mieć również dowolne inne formaty.

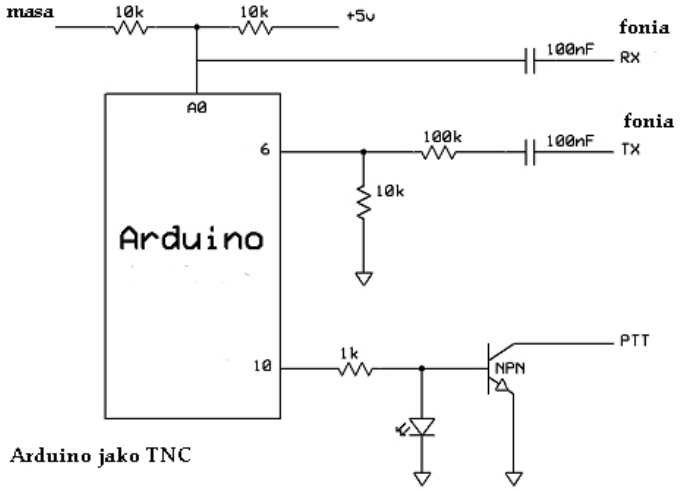
Fot. 4.1.1. Moduł APRS obsługujący kanał radiowy i dostęp do internetu – WX3in1. Moduł współpracuje z niektórymi modelami fabrycznych stacji meteorologicznych

Fot. 4.1.2. Bramka internetowa APRS

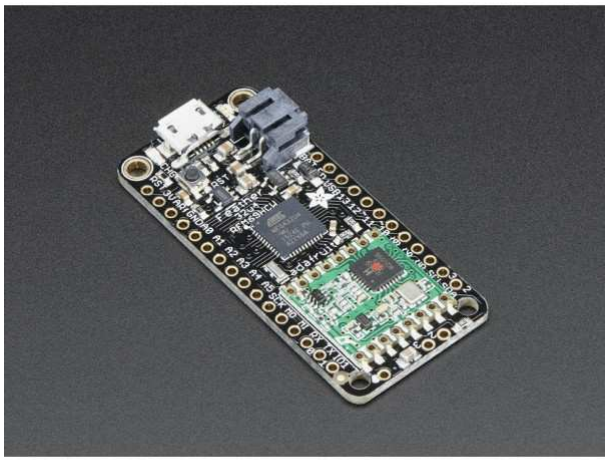


Fot. 4.1.3. Nowoczesny modem TNC produkcji firmy SCS-PTC

Przykładem wykorzystania „Arduino” jako TNC jest konstrukcja M0PZT opisana w witrynie *www.m0pzt.com*.



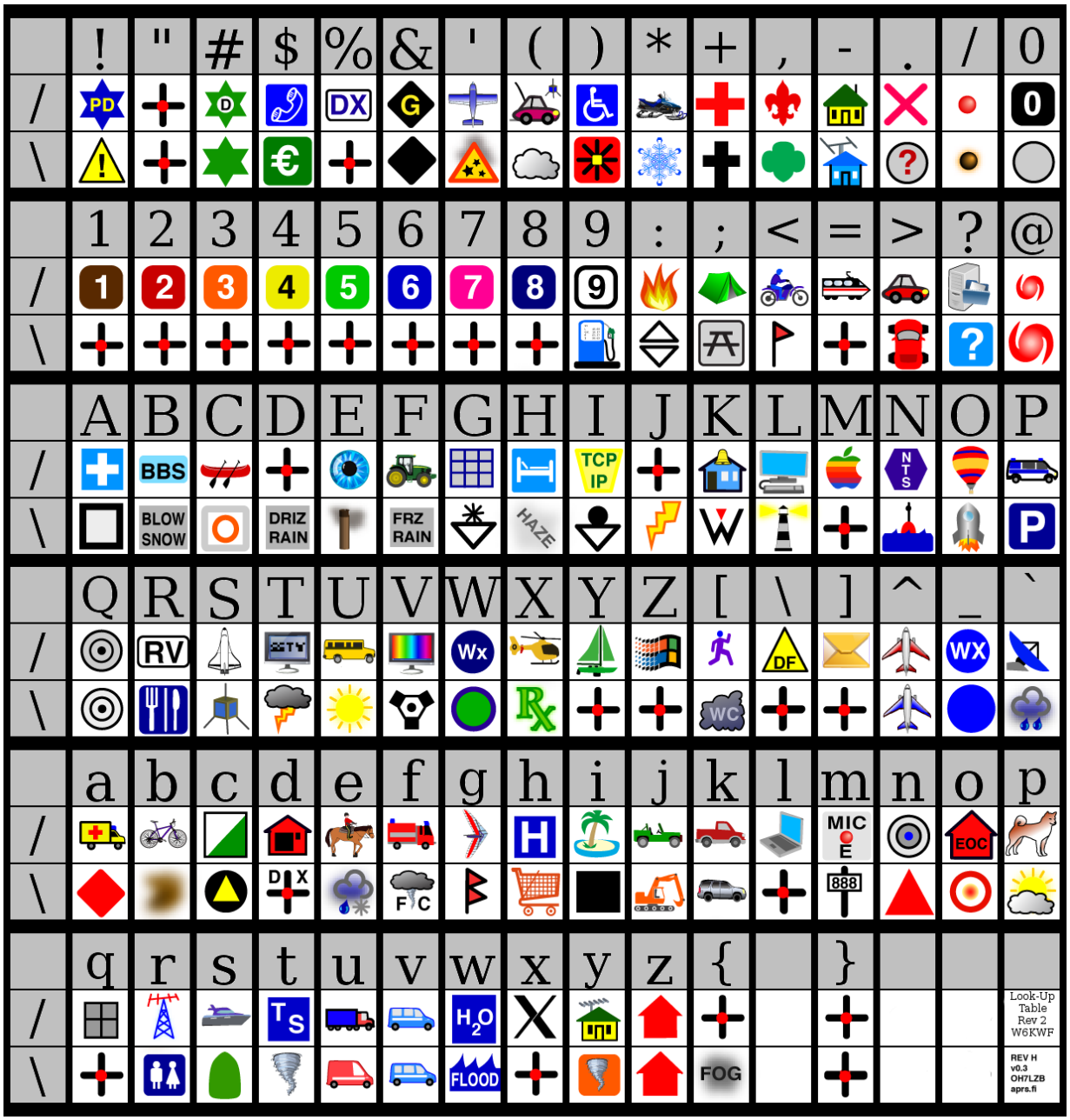
Rys. 4.1.4. Arduino jako TNC



Fot. 4.1.5. Moduł packet radio z RFM69HCW

Moduł „Adafruit Feather 32u4” z RFM69HCW pracuje emisją packet-radio w paśmie 433 MHz. Oprócz niego produkowane są też moduły dla pasm 868 MHz lub 915 MHz. Jest on wyposażony w mikroprocesor Atmega 32µ4 zasilany napięciem 3,3 V i taktowany z częstotliwością 8 MHz. Dla packet-radio istnieje specjalna biblioteka programów do niego. Moce wyjściowe nadajnika leżą w zakresie +13 do +20 dBm. Płytka ma wymiary 51 x 23 x 8 mm i masę 5,5 g.

Rys. 4.1.6. Komunikat meteorologiczny APRS na tle mapy z witryny *aprs.fi*



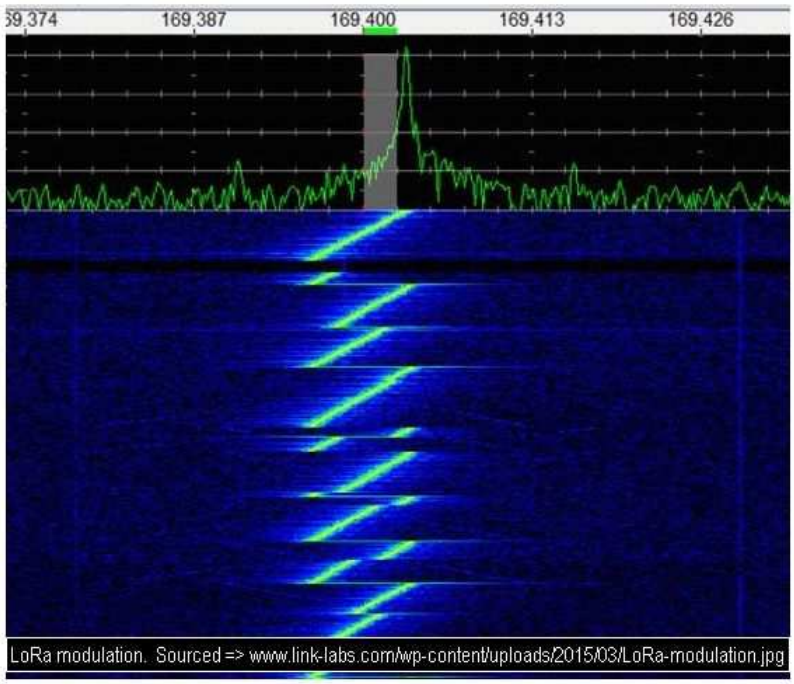
Rys. 4.1.7. Tabele symboli stacji APRS. Elementy tabeli pierwszej są poprzedzone znakiem „/”, a drugiej – znakiem „\”

System „LoRa” z rozpraszaniem widma

Oznaczenie „LoRa” jest skrótem od „Long Range” mającego oznaczać telemetrię dalekosiężną, przy czym o ile w porównaniu z rozpowszechnionymi systemami transmisji danych pomiarowych na dystansach kilku lub kilkudziesięciu metrów zasięgi dochodzące do 20 lub więcej km można uznać za dalekie. W oczach krótkofalowców nie są to oczywiście żadne DX-y, ale przecież nie o to najbardziej chodzi.
System „LoRa” korzystający z transmisji z rozpraszaniem widma sygnału (ang. *spread spectrum*) został opracowany w 2013 roku we francuskiej firma „Cycleo” wykupionej następnie przez firmę „Semtech”. Obecnie na rynkach dostępne są moduły dla pasm 433 i 868 MHz, a w USA także pracujące w paśmie 915 MHz. Do zastosowań krótkofalarskich nadają się oczywiście moduły na zakres 433 MHz (typu RFM98W-433S2 itp.), a do zastosowań nie wymagających licencji amatorskiej – moduły dla pasma 868 MHz (RFM95W-868S2). W paśmie 868 MHz w zależności od podzakresu obowiązuje ograniczenie mocy do 10 lub 25 mW i czasu nadawania do 0,1% lub 1% czasu pracy – liczonego w skali godzinowej.
Pomimo niskich mocy nadawania – od kilkunastu do maksimum 120 mW – osiągalne są dzięki zastosowaniu transmisji z rozpraszaniem widma stosunkowo dalekie zasięgi. W stosunku do transmisji z kluczowaniem FSK lub GMSK są one około dziesięciokrotnie wyższe. Zależny od współczynnika rozpraszania widma zysk systemowy wynosi tutaj około 20 dB.
Niskie moce nadawania pozwalają z kolei na bateryjne zasilanie urządzeń. Obecnie produkowane moduły pobierają przy nadawaniu od 20 mA przy mocy 5 mW do 120 mA przy mocach rzędu 120 mW. Krotkofalowcy mogą, w odróżnieniu od użytkowników nielicencjonowanych korzystać z dowolnych anten zewnętrznych i nie obowiązują ich ograniczenia czasu nadawania.
Dopuszczalne szybkości transmisji wynoszą wprawdzie od 180 bit/s do 37,5 kbit/s ale dla uzyskania możliwie największej czułości stosowane są szybkości z dolnego zakresu, co praktycznie jest w pełni wystarczające ponieważ ilość transmitowanych danych jest raczej niewielka. Dla minimalnych szybkości osiągane są czułości -148 dBm, natomiast przykładowo dla 1200 bit/s już tylko -119 dBm.
Możliwa jest też automatyczna zmiana szybkości transmisji w zależności od jakości łącza (ang. *adaptive data rate – ADR*).
Maksymalna długość transmitowanego pakietu wynosi 256 bajtów, wliczając w to dane korekcyjne FEC. Jest ona ograniczona dlugością rejestru przesuwnego zawartego w obwodzie scalonym modemu.
Współczynnik rozpraszania widma (stosunek szerokości pasma sygnału nadawanego do szerokości widma kanału podstawowego) leży w granicach 6 – 12, a pasmo sygnału transmitowanego zajmuje od 7,8 do 500 kHz. Standardowo stosowane jest pasmo 125 kHz.
Moduły radiowe „LoRa” są stosowane w profesjonalnych systemach telemetrycznych, w zdalnym odczycie liczników elektrycznych i innych przyrządów, w systemach bezpieczeństwa itp.
Do spraw problematycznych należy wprawdzie kwestia zabezpieczenia transmitowanych danych ale w zastosowania amatorskich sprawa ta nie ma większego znaczenia.
Moduły „LoRa” pozwalają dodatkowo na transmisję danych z kluczowaniem FSK, GFSK, MSK, GMSK i OOK.
Do najczęściej stosownych metod rozpraszania widma sygnału należą kluczowanie częstotliwości (ang. *frequency hopping – FH*), kluczowanie fazy kodem losowym (ang. *direct sequence spread spectrum – DSSS*) i linowa zmiana częstotliwości. Do rozpraszania widma stosowane są kody pseudoprzypadkowe. W najprostszym przypadku są one generowane za pomocą rejestrów przesuwnych z odpowiednio dobranymi sprzężeniami zwrotnymi, ale mogą być także generowane programowo. Istnieją całe rodziny takich kodów, a nowe kody mogą być też generowane przez kombinację istniejących. Teoria kodów pseudoprzypadkowych jest dość rozbudowana i skomplikowana.
Zysk systemowy jest zależny od stopnia rozproszenia widma czyli stosunku szerokości pasma sygnału rozproszonego do sygnału podstawowego. Sygnał o rozproszonym widmie jest odbieralny przez klasyczne odbiorniki wąskopasmowe jako szum. Dodaje się on do szumów pochodzących z innych źródeł. Przy wielu zachodzących na siebie sygnałach o rozproszonym w dowolny sposób widmie selektywny wybór możliwy jest dzięki stosowaniu różnych kodów. Dla prawidłowego odbioru konieczna jest

oprócz zgodności kodów także zgodność ich faz (w sygnale odbieranym i w kodzie generowanym w odbiorniku), a także zgodność ich częstotliwości zegarowych. Sprawa zapewnienia synchronizacji należy do najważniejszych problemów w komunikacji z rozpraszaniem widma sygnału.
Oprócz radiokomunikacji transmisja z rozpraszaniem widma jest stosowana m.in. w zdalnym sterowaniu i w technice pomiarowej. Pomiar przesunięcia kodu odbieranego w stosunku do nadawanego pozwala, zwłaszcza przy użyciu odpowiednio długich kodów, na dokładne pomiary odległości.
W modułach „LoRa" stosowany jest uproszczony system liniowej zmiany częstotliwości (ang. *chirp – CSS*).

Rys. 4.2.1. Transmisja sygnału o rozproszonym widmie. Sygnał użytkowy jest po stronie nadawaczej rozpraszany, a po stronie odbiorczej skupiany w identyczny sposób i przy użyciu identycznego kodu dla odzyskania danych w paśmie podstawowym. Ewentualnie odebrane sygnały wąskopasmowe znajdujące się w tym samym paśmie ulegają w odbiorniku rozproszeniu i nie powodują zakłóceń



Rys. 4.2.2. Przebieg sygnału systemu „LoRa" w funkcji czasu i częstotliwości (źródło: dokumentacja „LoRa")

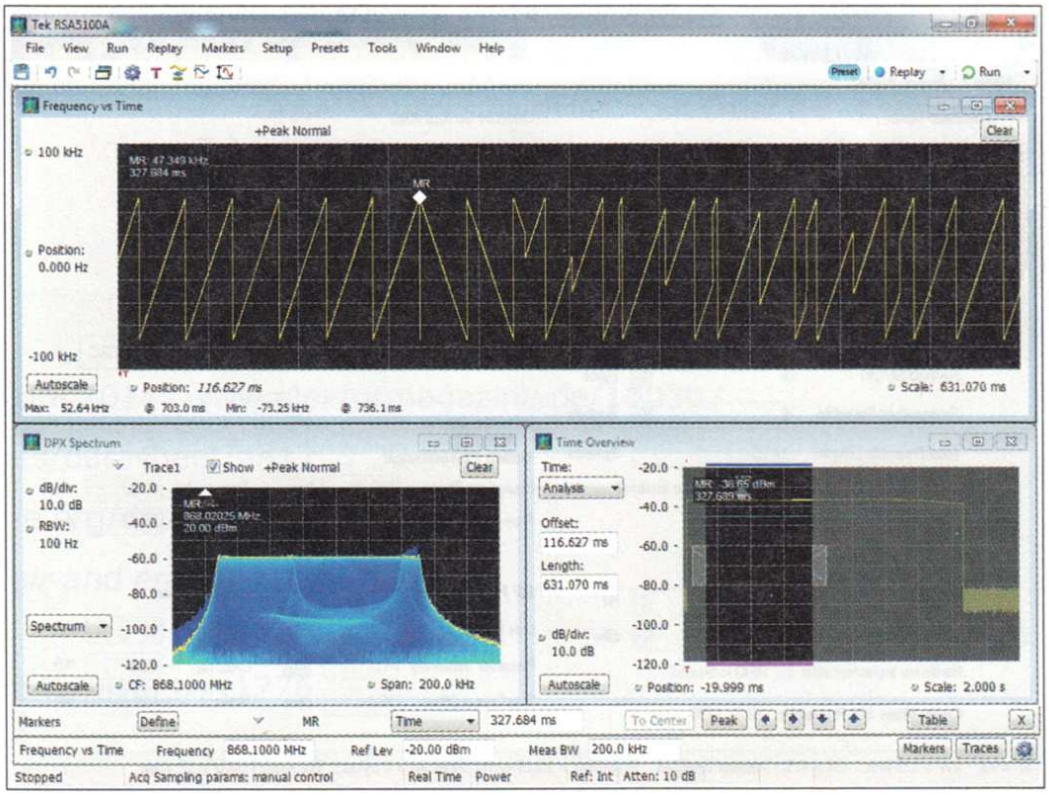
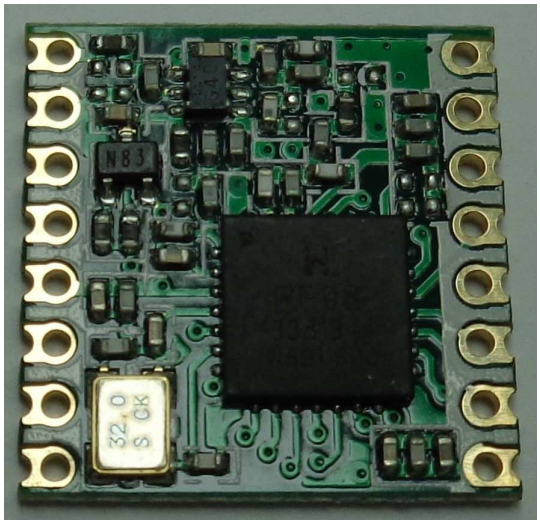


Bild 4. Ein LoRa-Frame, dargestellt von einem Real-time-Spektrumanalyser, hier ein Tektronix RSA5106.

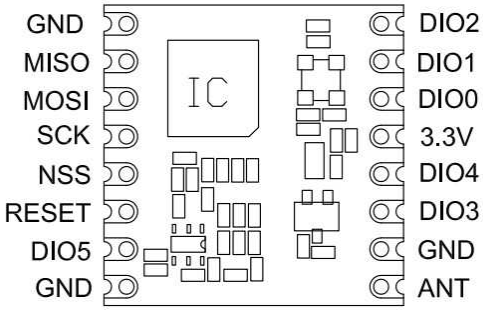
Rys. 4.2.3. Sygnał systemu „LoRa” zmierzony analizatorem widma RSA5106 firmy Tektronix (źródło: *Elektor* maj-czerwiec 2017)

Na ilustracji 4.2.3. Przedstawiona jest dokonana w laboratorium *Elektora* analiza sygnału systemu „LoRa” dla pasma o szerokości 125 kHz, współczynnika rozpraszania 12, FEC 4/5 i dewiacji również 125 kHz. W bloku synchronizacyjnym przebiegi są odwrócone: częstotliwość zmniejsza się, a nie wzrasta w funkcji czasu. Blok synchronizacyjny składa się z 8 symboli, przy czym do synchronizacji odbiornika wystarczają 4 z nich.
Bity zgrupowane są w symbole 12-bitowe – przyjmują więc wartości od 0 do 4095. Czas trwania symbolu wynosi 32,8 msek. Przepustowość brutto równa jest 366,2 bit/s, co przy FEC 4/5 daje przepustowość netto 292,9 bit/s.
Maksymalna dewiacja 125 kHz jest podzielona na 2^{12} odcinków: 125000/4096 = 30,52 Hz. Dewiacja chwilowa równa jest wartości symbolu x 30,52 Hz. Transmisja 8 bajtów użytkowych trwa 925 msek.
Najczęściej system jest stosowany do transmisji niewielkich ilości danych i to niezbyt często. Transmisja z dużymi szybkościami się nie opłaca. Oprócz zastosowań naziemnych w praktyce amatorskiej interesująca jest transmsja danych z balonów. Dzięki znacznym wysokościom lotu uzyskuje się duże zasięgi stacji. Inną interesującą dziedziną zastosowań może być śledzenie zwierząt wyposażonych w nadajniki.
Moduły RFM98W i podobne komunikują się z mikrokomputerem za pomocą złącza SPI (*Serial Peripherial Interface*), w którego skład wchodzą sygnały MOSI (*Master Output Slave Input*), MISO (*Master Input Slave Output*), SCK (sygnał zegarowy) i SS (*Slave Select*). Sygnał selekcji urządzenia SS musi być połączony z wejściem NSS modułu „LoRa”. Jako wyjście selekcyjne po stronie Arduino może służyć dowolne wyjście logiczne.
Kontakty DIO0 – DIO5 modułu są wykorzystywane przez bibliotekę LMIC do różnych celów j.np. informacji o pracy urządzenia.

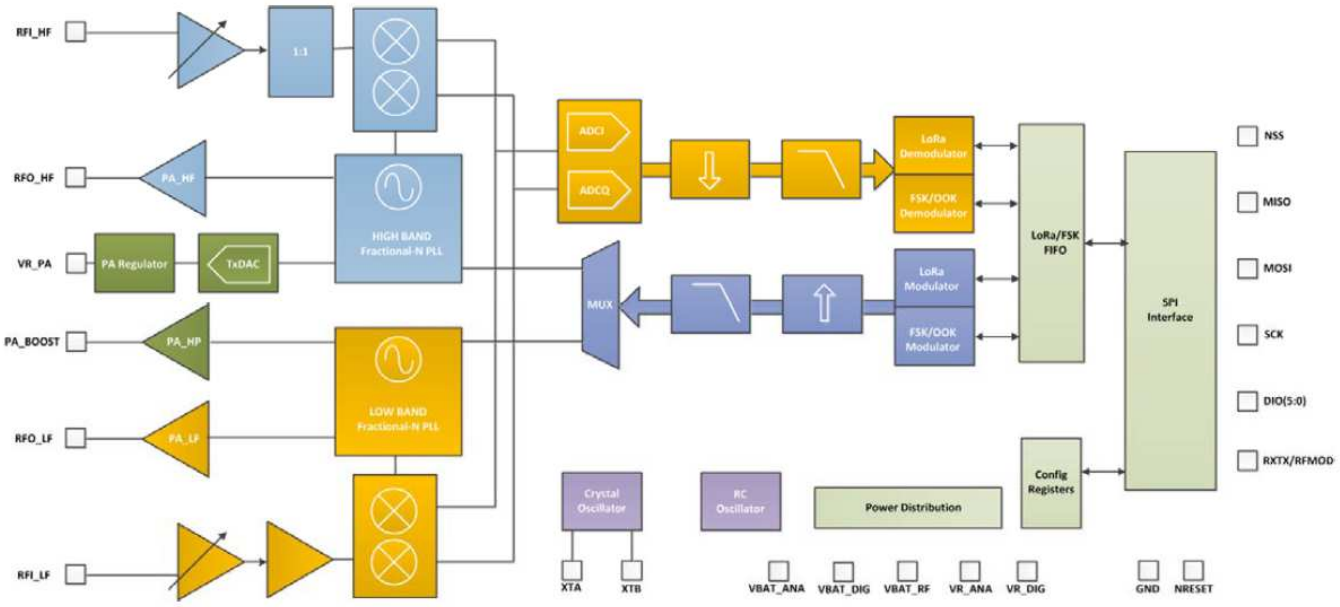
.



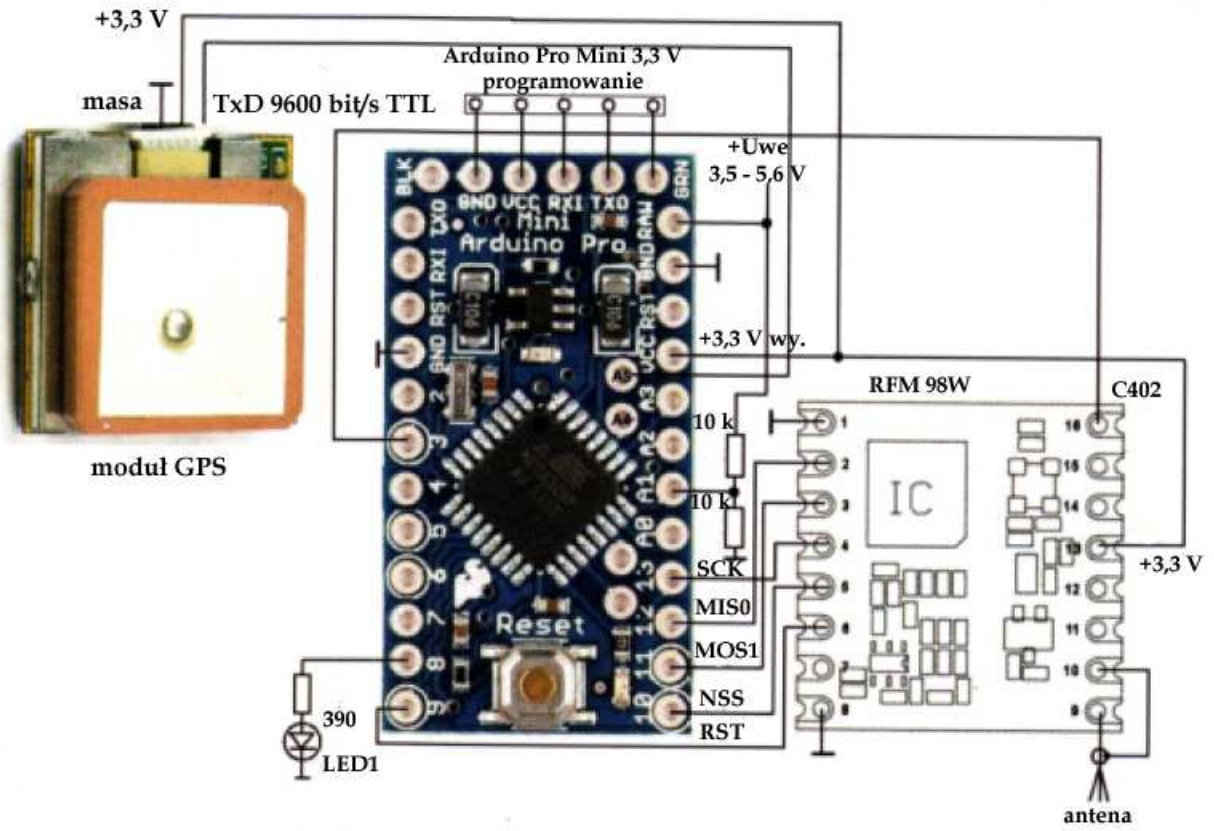
Fot. 4.2.4. Moduł dla pasma 433 MHz…



Rys. 4.2.5. …i jego wyprowadzenia

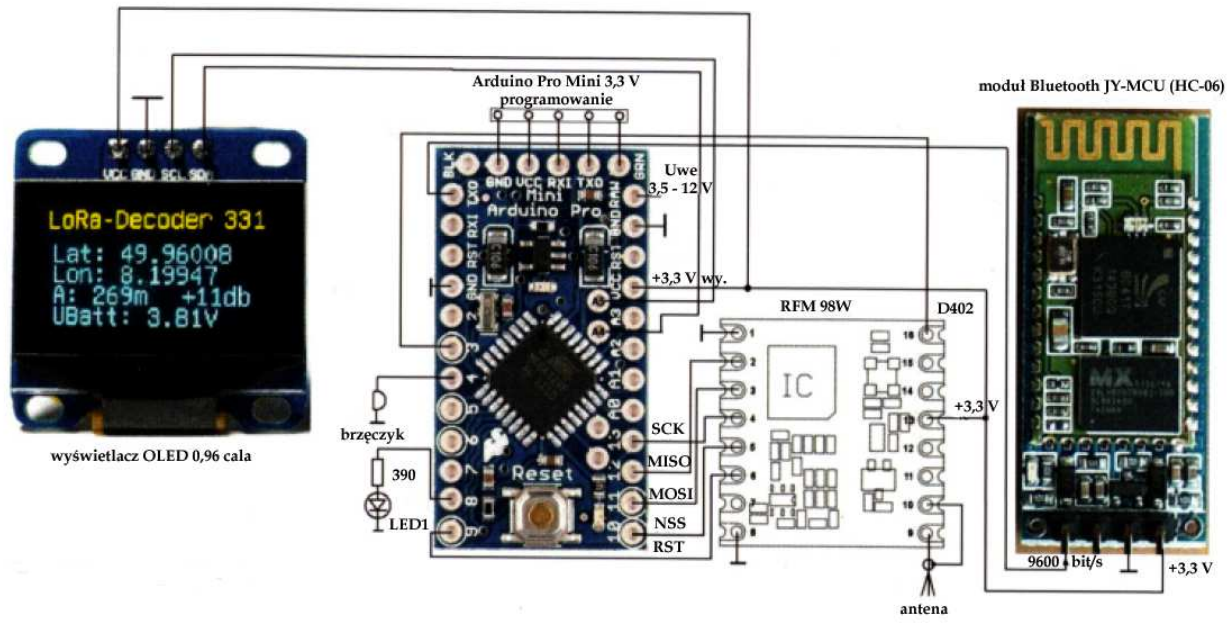


Rys. 4.2.6. Schemat blokowy modułu



Rys. 4.2.7. Nadajnik APRS z modułem „LoRa” (źródło: [4] i QSP 4/2017)

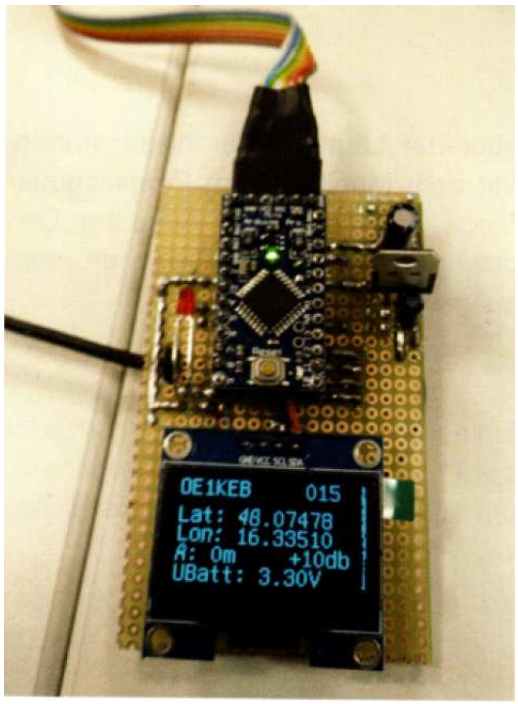
W przedstawionym na ilustracji 4.2.7 nadajniku APR zastosowano Arduino Mini zasilany napięciem 3,3 V co ułatwia połączena obu części, a także modułu odbiornika GPS. Oprócz danych APRS nadawana jest informacja o napięciu zasilania. Napięcie to jest doprowadzone do wejścia A1 „Arduino” przez dzielnik 1:2 złożony z dwóch oporników 10 kΩ. Krótkofalowcy eksperymentowali również z transmisją danych meteorologicznych: ciśnienia atmosferycznego, temperatury i wilgotności.



Rys. 4.2.8. Odbiornik APRS (źródło: [4] i QSP 4/2017)

W pasującym do tego odbiorniku APRS zastosowano wyświetlacz OLED połączony z mikrokomputerem za pomocą magistrali I2C (sygnał zegarowy SCL pochodzi z wyjścia A5 „Arduino”, a sygnał danych SDA – z wyjścia A4). Moduł Bluetooth służy do połączenia odbiornika z komputerami lub telefonami komórkowymi. Dla telefonów i komputerów androidowych dostępne są m.in. programy *BluTerm, W2APRS* i *Lotus Map Pro*. Pierwszy z nich jest programem terminalowym pozwalającym na odczyt danych w postaci tekstowej, a następne umożliwiają przedstawienie pozycji odbieranej stacji na mapie.

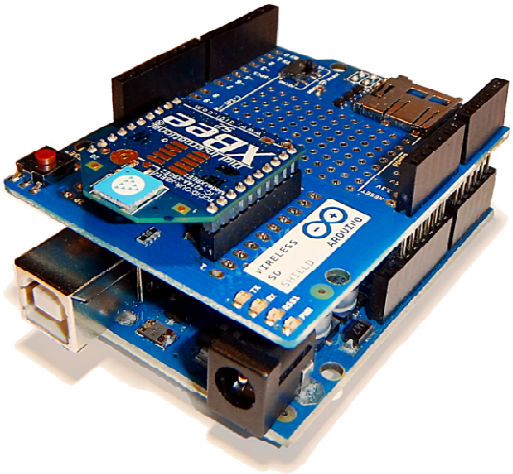
Fot. 4.2.9. Transmisja danych meteorologicznych: ciśnienia, temperatury, wilgotności względnej i wysokości zmierzonych za pomocą czujnika BME280 firmy „Bosch”. Dodatkowo wyświetlana jest informacja o sile odbieranego sygnału (odstępie poziomu sygnału od szumu)



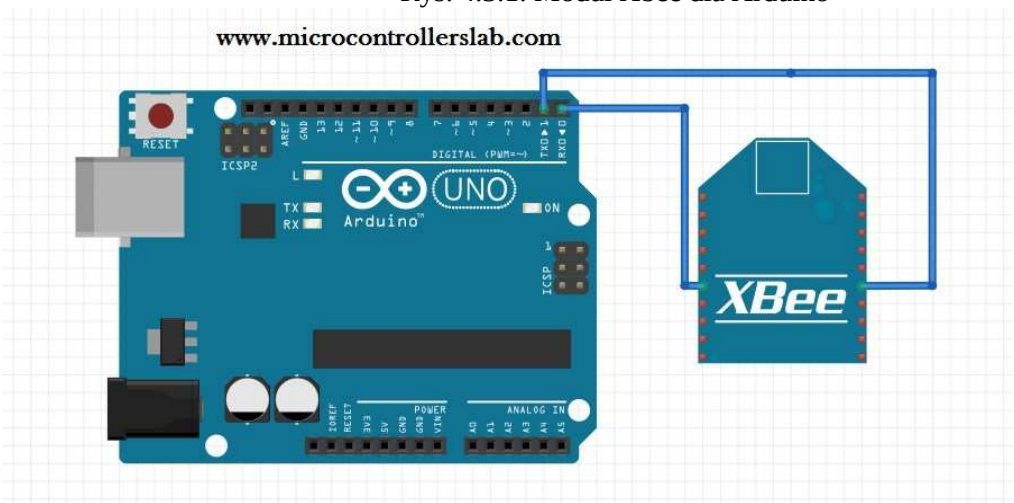
Fot. 4.2.10. Odbiornik konstrukcji OE1KEB (źródło: QSP 4/2017)

XBee

Pomimo łatwej dostępności modułów Xbee (zgodnych ze standarden ZigBee opartym na normie IEEE 802.15.4) dla „Arduino” rozwiązanie to jest znacznie mniej atrakcyjne. Łączność radiowa odbywa się w ogólnie dostępnym paśmie ISM 2,4 GHz co przy mocach nadajników rzędu miliwatów i znacznie większym tłumieniu fali aniżeli w zakresie 433 MHz ogranicza zasięgi do 30 m wewnątrz budynków i do 90–100 m na zewnątrz. Produkowane są również modele o mocach do 63 mW i zasięgach 90 m wewnątrz budynków, a 1600 m ma zewnątrz. Przy napięciu zasilania 3,3, V pobór prądu wynosi 250 mA przy nadawaniu z mocą 18 dBm, a 55 mA przy odbiorze.
Szybkości transmisji leżą w zakresie 1200 bit/s – 250 kbit/s.
W niektórych rozwiązaniach eksperymentalnych, zwłaszcza szkolnych może okazać się to jednak interesującą możliwością. Protokół Xbee jest dostosowany do pracy w sieciach o różnych topologiach: gwiaździstej, hierarchicznej lub sieci stacji równouprawnionych.



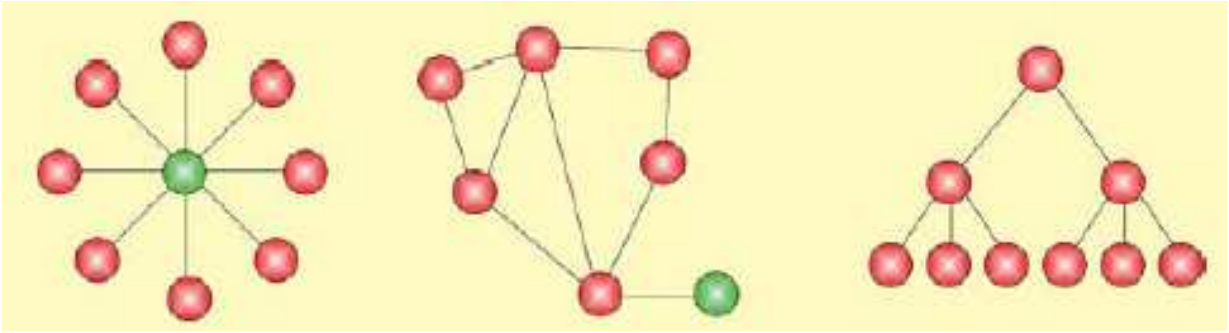
Rys. 4.3.1. Moduł Xbee dla Arduino



Rys. 4.3.2. Moduł Xbee jest połączony ze złączem szeregowym (kontaktami RXD i TXD) „Arduino” i zasilany napięciem 3,3 V (źródło: *www.microcontrollers.lab*)

Obsługa Xbee w programie (przykład):

```
void setup()
  {
    Serial.begin(9600);
  }
void loop()
  {
    while (Serial.available() ) {
      Serial.write(Serial.read()); /
    }
  }
```



Rys. 4.3.3. Możliwe topologie sieci

Moduły TRM-868-EUR

Moduły (należące do rodziny WiSE – *Wireless Serial Engine*) stanowią w praktyce radiowe przedłużenie złączy RS-232, RS422 i RS485 pracujące w paśmie 868 MHz. Do dyspozycji są albo dwa kanały szerokopasmowe o szerokości 600 kHz (868,300 MHz i 868,950 MHz) albo sześć wąskopasmowych o szerokości 200 kHz (868,225 – 869,850 MHz). Dopuszczalne przepływności w kanałach wąskopasmowych wynoszą 9600 bit/s, a w kanałach szerokopasmowych – 76,8 kbit/s. Zasilane napięciem 2,7 – 3,6 V moduły są dołączane do złącza szeregowego (UART) komputerów lub mikrokomputerów. Wykorzystywane są sygnały RxD, TxD i CTS z poziomami napięć CMOS. Maksymalne moce nadawania wynoszą 13 – 15 dBm przy użyciu dowolnych anten o oporności 50 Ω. Pobór prądu przy nadawaniu wynosi 26 – 65 mA, a przy odbiorze 16 – 20 mA.
Do rozpoznawania przekłamań transmisji służy 16-bitowa suma kontrolna CRC.

Mikrokomputery i urządzenia pomocnicze

Mikrokomputery

Arduino

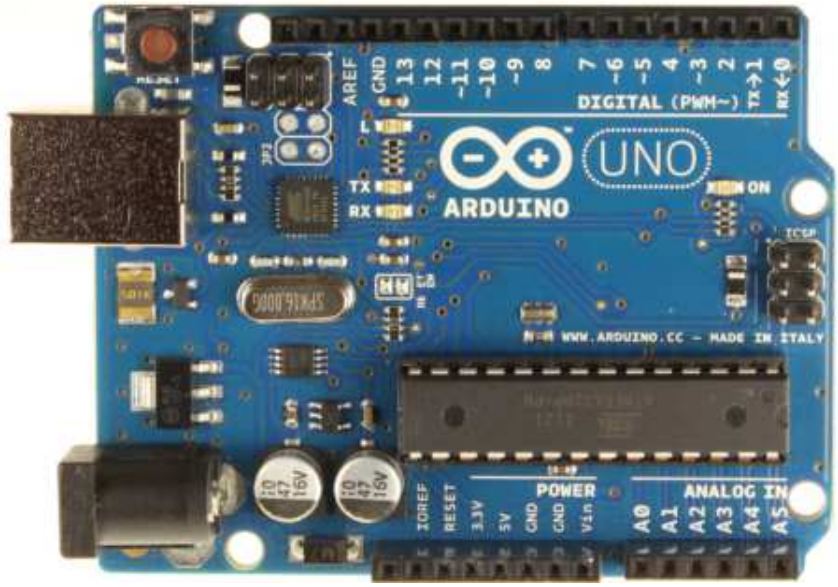
Arduino jest nieskomplikowanym i niedrogim mikrokomputerem jednopłytkowym opartym na mikroprocesorach z rodziny Atmega. Najczęściej używane są obecnie mikroprocesory Atmega328, Atmega 32µ i Atmega2560. W zależności od modelu mikrokomputera i użytego w nim procesora pamięci programu (EPROM) mają pojemności 16 – 512 kB, pamięci robocze (RAM) 1–96 kB, a pamięci nieulotne EEPROM 0,512 – 4 kB. Z zapisu i odczytu pamięci EEPROM mogą korzystać pracujące na mikrokomputerze programy dlatego też obszar ten jest przeznaczony w pierwszym rzędzie dla rzadko zmieniających się danych j.np. parametrów konfiguracyjnych czy kalibracyjnych w układach pomiarowych. Dane te nie ulegają skasowaniu po wyłączeniu zasilania ale ponieważ liczba cykli zapisu jest stosunkowo ograniczona nie należy wykorzytywać tej pamięci jako pamięci roboczej.
Większość modeli posiada kilka wejść analogowych, kilka wyjść sygnałów z modulacją szerokości impulsów (ang. *PWM*) i kilkanaście konfigurowalnych wejść/wyjść logicznych (patrz tabela 1.1). Są one przeważnie doprowadzone do listew kontaktowych umieszczonych na krawędziach płytki. Konstrukcja ta pozwala na wtykanie do nich modułów rozszerzeń zwanych w literaturze angielskiej *shield* czyli w dosłownym tłumaczeniu – tarcza. W gwarze Arduino programy noszą angielską nazwę *sketch* czyli szkic, my jednak pozostaniemy przy polskiej terminologii.
Większość modeli jest wyposażona w gniazdo USB wykorzystywane do połączenia z komputerem PC na czas ładowania i uruchamiania programów. Gniazdo to może służyć także do zasilania mikrokomputera napięciem 5 V chociaż dodatkowo możliwe jest zasilanie przez oddzielne gniazdo napięciami 7–9 lub 7–12 V. W modelach Leonardo i DUE do gniazd USB można podłączyć komputerowe klawiatury i myszy. W modelach nieposiadających gniazda USB do ładowania programu służy przeważnie złącze ICSP. Złącze to wykorzystywane jest także do ładowania programów skompilowanych przez kompilatory innych języków.

Tabela 5.1.1. Parametry techniczne wybranych modeli Arduino

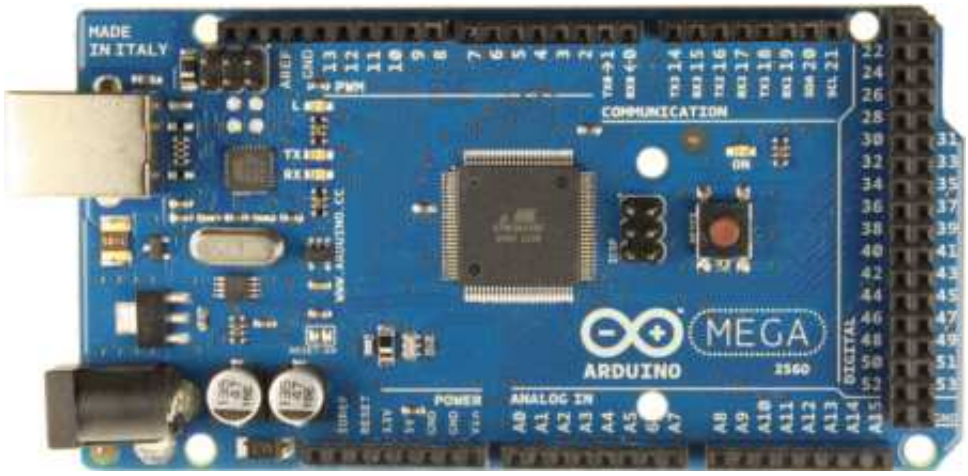
	Arduino UNO	**Arduino Leonardo**	**Arduino DUE**	**Arduino MEGA 2560**	**Arduino Nano**
Procesor	Atmega328	Atmega32µ4	AT91SAM 3X8E	Atmega2560	Atmega328
Takt	16 MHz	16 MHz	84 MHz	16 MHz	16 MHz
Wejścia analogowe	6	12	12	16	6
Wyjścia analogowe	0	0	2	0	0
We./wy. logiczne	14	20	54	54	14
Wyjścia PWM	6	7	12	15	6
Pamięć programu	32 kB	32 kB	512 kB	256 kB	32 kB
Pamięć robocza RAM	2 kB	2,5 kB	96 kB	8 kB	2 kB
Pamięć EEPROM	1 kB	1 kB	-	4 kB	1 kB
UART	1	1	4	4	1
Gniazdo USB	standard B	mikro	2 mikro	standard B	mini B
Zasilanie	5 / 7–12 V	5 / 7–12 V	3,3 / 7–12 V	5 / 7–12 V	5 / 7–9 V

Mikrokomputery Arduino zostały opracowane z myślą o użytkownikach mających niewielkie doświadczenie w programowaniu (przykładowo osobach związanych ze sztuką i pragnących osiągnąć różne efekty dźwiękowe lub optyczne) albo do celów dydaktycznych i dlatego zarówno język programowania jest stosunkowo prosty jak i środowisko programistyczne jest łatwe w instalacji i obsłudze. W modelach wyposażonych w gniazdo USB programy są ładowane do pamięci procesora za jego pośrednictwem bez konieczności korzystania z oddzielnego programatora. W tym celu procesory zamontowane na płytkach „Arduino” posiadają wprowadzony program ładujący (ang. *bootloader*) zajmujący ok. 0,5 – 2 kB

pamięci programu. Możliwe jest także ładowanie programu bezpośrednio przez znajdujące się na płytce złącze ICSP ale bywa to w praktyce rzadziej stosowane. Może ono służyć przykładowo do ładowania programów skompilowanych przez inny dowolny kompilator np. języka C dla procesorów AVR. Konieczne jest wówczas użycie dodatkowego programatora co oznacza dalsze inwestycje.
Generator częstotliwości zegarowej jest stabilizowany kwarcem dzięki czemu jest ona wystarczająco stabilna i dokładna dla większości zastosowań.
Zestawienie najważniejszych parametrów technicznych najczęściej używanych modeli zawiera tabela 5.1.1. Bieżące informacje o dostępnych modelach mikrokomputerów, bibliotekach programów, aktualnych wersjach środowiska programistycznego, wiele praktycznych przykładów programów, porad itp. zawiera witryna *www.arduino.cc*.



Fot. 5.1.1. Arduino UNO



Fot. 5.1.2. Arduino MEGA2560

Tabela 5.1.2. Sygnały na złączach płytki „Arduino UNO” i lewej części „Arduino MEGA2560”

Kontakt	Sygnały
Logiczny 0	Złącze szeregowe RX, połączony też z kontrolerem FTDI.
Logiczny 1	Złącze szeregowe TX, połączony też z kontrolerem FTDI.
Logiczne 2, 3	Zewnętrzne przerwania jeśli zostały włączone za pomocą funkcji attachInterrupt(); reagują na poziom 0, zbocza sygnału lub zmianę wartości.
Logiczne 3, 5, 6, 9, 10, 11 (zaznaczone ~)	Wyjścia impulsów o modulowanej szerokości (PWM) – wyjścia pseudo-analogowe.
Logiczny 4	Sterowanie dostępem do modułów pamięci SD. Wymagana dodatkowa biblioteka „SD”.
Logiczne 10, 11, 12, 13	Sygnały dla złącza SPI – 10: SS, 11: MOSI, 12: MISO, 13: SCK; wymagana dodatkowa biblioteka funkcji SPI; połączone ze złączem ICSP.
Analogowe A4, A5	Sygnały dla złącza I2C – A4: SDA, A5: SCL; wymagana dodatkowa biblioteka „Wire”.
Logiczny 13	Dioda świecąca na płytce.
Analogowe A0 – A5	Wejścia analogowe, rozdzielczość 10 bitów (zakres wartości 0–1023), maksymalne napięcie wejściowe 5 V; w razie potrzeby stosować dzielniki napięć i zabezpieczenia przed przepięciami mogącymi uszkodzić procesor.
AREF	Napięcie odniesienia dla przetworników analogowo-cyfrowych; do przełączania służy funkcja analogReference().
GND	Masa
5 V	Stabilizowane napięcie +5 V
3,3 V	Stabilizowane napięcie 3,3 V
Vin	Niestabilizowane napięcie zasilania z gniazda koncentrycznego.
Reset	Służy do zerowania procesora.

Uwagi:

- Zależnie od konfiguracji i użycia pomocniczych bibliotek niektóre z kontaktów mogą pełnić różne funkcje.
- Numery kontaktów używane w programach odpowiadają numerom wydrukowanym na płytce a nie numerom wyprowadzeń procesora. Wyprowadzenia procesorów Atmega8, Atmega 168 i Atmega328 są identyczne. Główną różnicą między nimi są pojemności pamięci.

Poziomy napięć na wejściach i wyjściach logicznych odpowiadają standardowi TTL. Dopuszczalna obciążalność prądowa wyjść wynosi 40 mA. Należy jednak zwrócić uwagę aby przy maksymalnym obciążeniu większej liczby wyjść nie przekroczyć dopuszczalnej mocy strat procesora. Dla uniknięcia tej sytuacji korzystne jest dla większych obciążeń zastosowanie tranzystorów wykonawczych na wyjściach logicznych a wtórników napięciowych np. na wzmacniaczach operacyjnych na wyjściach analogowych. Arduino MEGA dysponuje znacznie większą liczbą wejść i wyjść każdego rodzaju.
Pracę „Arduino” w sieci lokalnej i dostęp do Internetu umożliwiają (opisane w tomach 20 i 21) moduły rozszerzeń Ethernetu i WiFi.
Dokładnymi odpowiednikami „Arduino UNO” i „Arduino MEGA 2650” są m.in. „Seeduino” i „Seeduino Mega”. Mikrokomputery „Arduino” są obecnie produkowane przez wiele firm, w tym przez AVT.

Malina

Mikrokomputer „Raspberry Pi” („Malina”) spotykany obecnie w postaci modeli 2 B i 3 B pracuje pod dostosowaną do jego możliwości odmianą Linuksa jest wyposażony w złącze wizyjne HDMI, cztery złącza USB, złącze dla modułów pamięci mikro SD, złącze Ethernetu, a model 3 także w modem WiFi. Ogólnie rzecz biorąc daje on znacznie więcej możliwości aniżeli „Arduino” ale w niektórych zastosowaniach możliwości takie nie są konieczne. Znaczna część programów jest tworzona w języku Python.

Fot. 5.1.3. „Malina 3 B

Tabela 1.1. Porównanie modeli „Maliny”

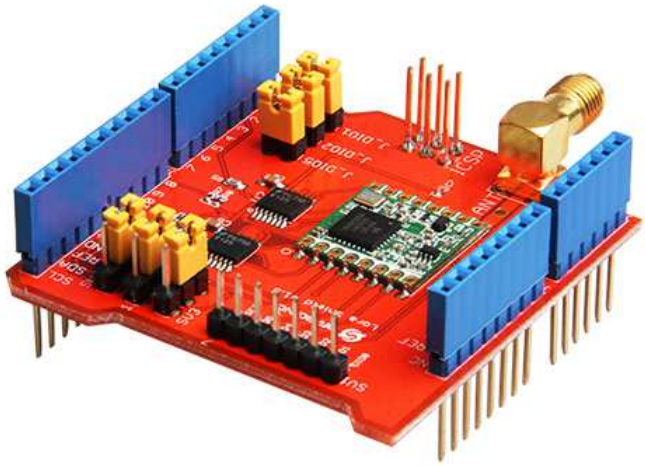
Parametr	3 Model B	2 Model B
Procesor	ARM Cortex-A53 (4x 1.2 GHz)	ARM Cortex A7 (4 x 900 MHz)
Pamięć RAM	1 GB	
Złącza USB	4 x USB 2.0	
Złacze Ethernet	10/100, gniazdo RJ45	
Pozostałe złącza	HDMI, wizyjne RCA, fonii 3,5 mm, mikroUSB (zasilanie do 2,4 A), GP10, wyświetlacz DSI, kamera CSI	
Pamięć programu	złącze dla mikroSD, 4 – 32 GB	
Złącze GPIO	40 kontaktów, w tym UART, I2C, SPI	
Modem	BCM43143 WiFi b/g/n	---
Bluetooth	4.1 (BLE)	---
System wizyjny	Dual Core VideoCore IV	
System operacyjny	najczęściej stosowany „Wheezy/Raspbian”, alternatywnie dostępne oddzielnie lub w zestawie „NOOBS”: Archlinux, Open ELEC, Piodora, RASP BMC, RISC OS	
Wymiary	85 x 56 x 17 mm	

Wymienione powyżej mikrokontrolery są jedynie najbardziej rozpowszechnionymi konstrukcjami, ale na rynku dostępnych jest szereg zbliżonych lub równoważnych rozwiązań, w tym „Banana Pi”, „Basic Stamp”, „C-Control” i wiele innych rozwiązań opartych m.in. o mikroprocesory AVR.

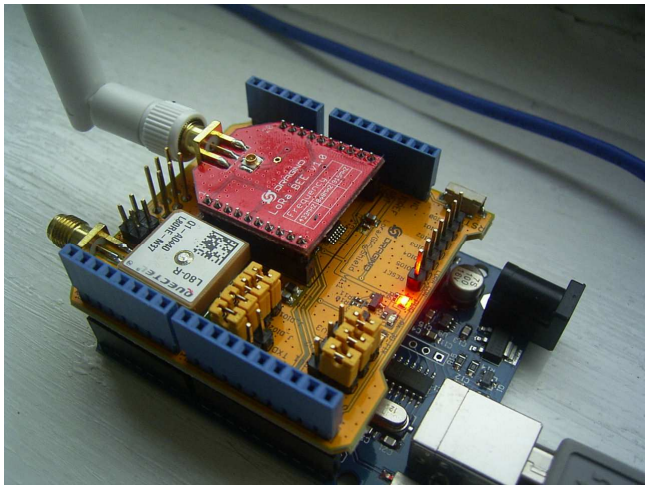
Rozszerzenia

Dragino

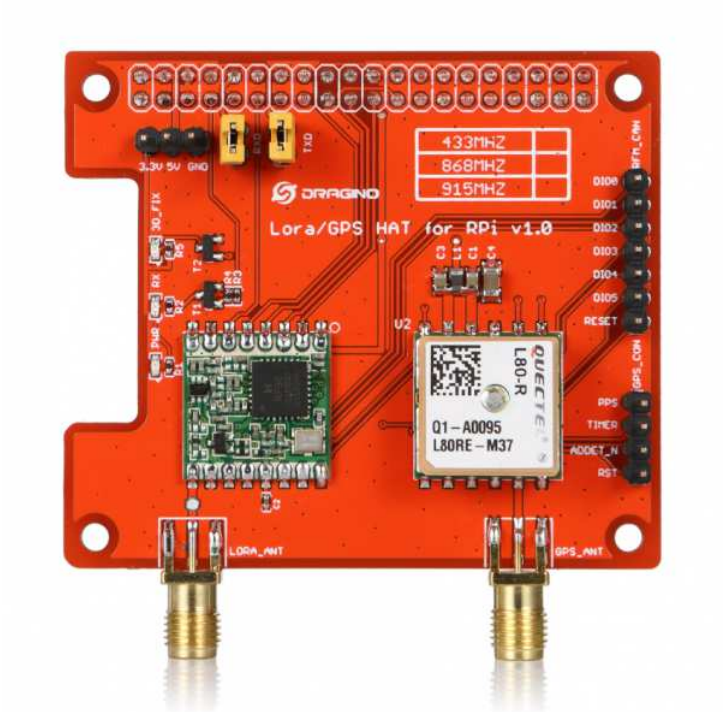
Płytki „Dragino” [5] są dodatkami (nakładkami) do „Arduina” lub „Maliny” zawierającymi moduły modemów „LoRa” i w niektórych modelach także odbiorniki GPS. Są one wyposażone w wersje modułów „LoRa” dla odpowiednich pasm. Dla pasma 433 MHz są to moduły RFM98-433S2, a dla 868 i 915 MHz – RFM95W-868S2 lub RFM95W-915S2. Nakładki znacznie ułatwiają korzystanie z technologii „LoRa” ponieważ odciążają konstruktorów od prac montażowych i wymagają jedynie połączenia ich z mikrokomputerami za pomocą pasujących listew wtykowych.



Fot. 5.2.1. Płytka „Dragino” dla „Arduino UNO, Mega, Due i Leonardo



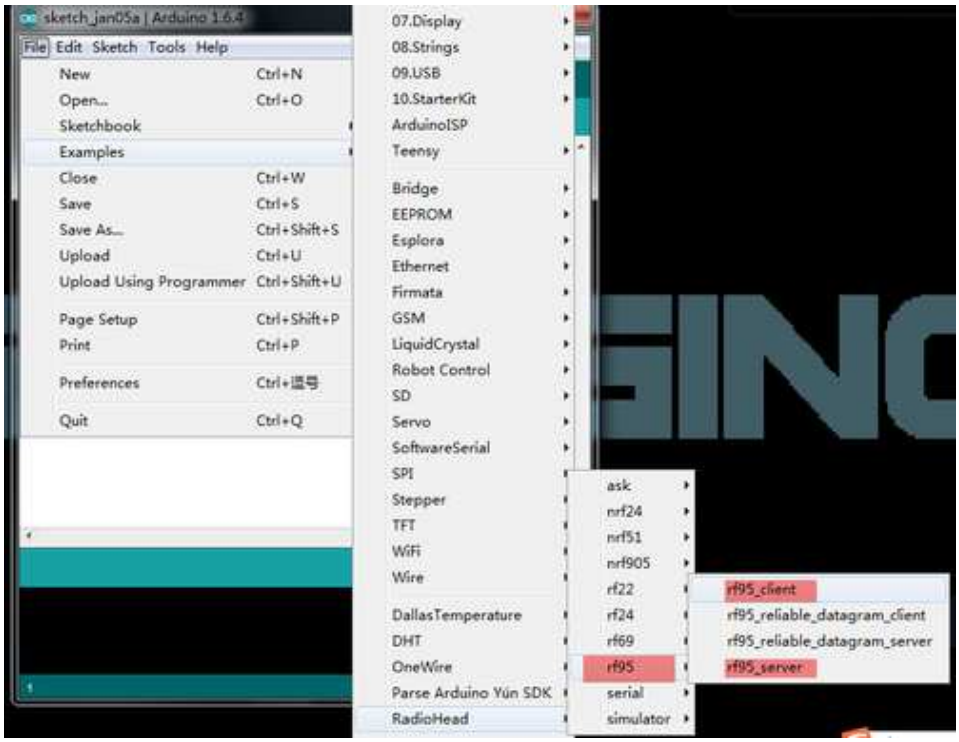
Fot. 5.2.2. Wersja z dodatkowym odbiornikiem GPS umieszczona na „Arduino UNO”.



Fot. 5.2.3. Płytka z odbiornikiem GPS przeznaczona dla „Maliny” 2 i 3. Po lewej stronie widoczny jest moduł :LoRa”, a po prawej – odbiornik GPS

Fot. 5.2.4. Płytka zamontowana na „Malinie”

Do ich obsługi konieczne są dla „Arduino” dodatkowe biblioteki. Krótkofalowcy korzystają przeważnie z dostępnej bezpłatnie na *github.com* biblioteki „RadioHead”. Tam też można znaleźć również przykładowe programy, w tym również dla „Malin”.
Dla modułów wyposażonych w modemy (transceiwery) SX1272, SX1276, HOPE RFM9x itp. opracowano dostosowane do możliwości „Arduino” biblioteki LMIC (LoRaMAC in C).



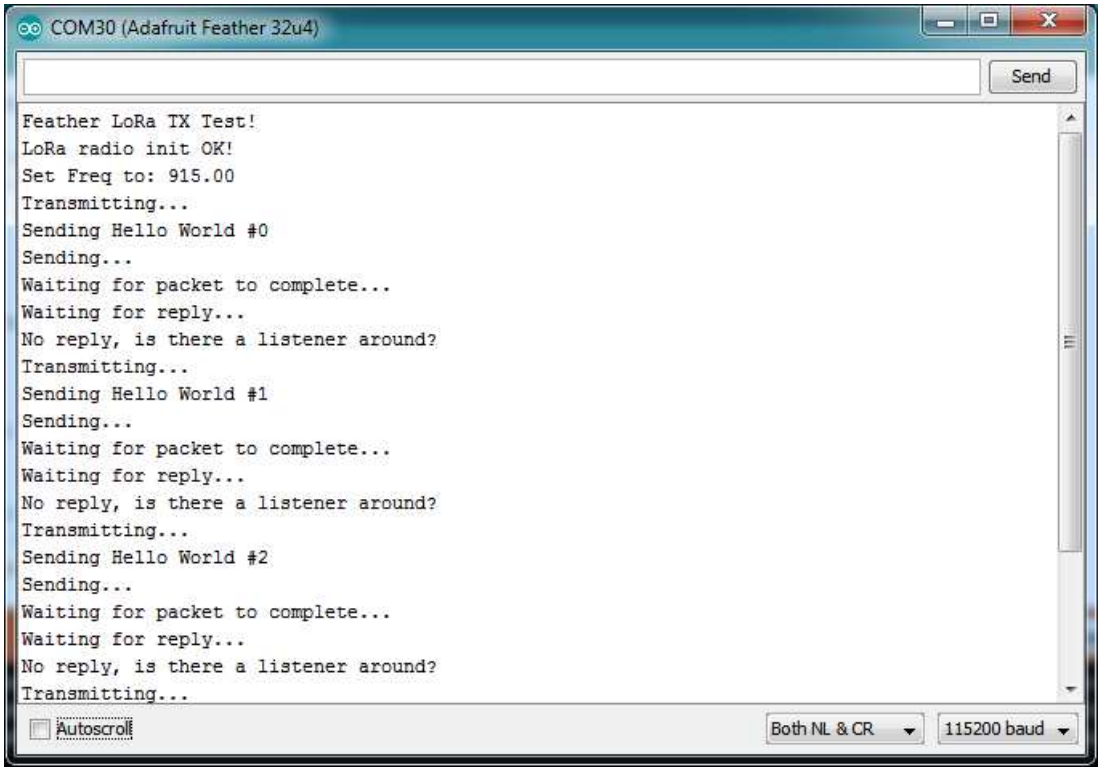
Rys. 5.2.5. Wywołanie przykładowych programów w środowisku programistycznym „Arduino”

Przykładowy program nadawczy dla „Arduino” z modułem „Dragino” i użyciem biblioteki „RadioHead” (źródło: Adafruit). W programie mozna wybrać również inna częstotliwość w granicach zakresu. Oczekiwanie na odpowiedź złącza szeregowego while (!Serial) można usunąć jeśli „Arduino nie jest połączony z komputerem. W opisie programu zalecane jest wprawdzie ustawienie najwyższej możliwej mocy nadawania (23 dBm) i obniżanie jej jeśli nie będzie konieczna, ale warto też zwrócić uwagę na to, czy moduł nie przegrzewa się i czy nie powoduje to płynięcia częstotliwości nadawania. Mogłoby to uniemożliwić dekodowanie sygnałow stacjom dostrojonym do częstotliwości nominalnej. Efekt ten jest oczywiście zależny od czasu nadawania czyli długości i częstotliwości powtarzania komunikatów. W poniższym programie komunikaty są nadawane co sekundę. W programie nie przewidziano żadnego adresowania i jeśli jest to konieczne należy go odpowiednio uzupełnić.

```
// LoRa 9x_TX
// -*- mode: C++ -*-
// Przykładowy program ilustrujący pracę prostego systemu nadawczego z modułami RH_RF95
// RH_RF95 nie posiadają możliwości adresowania i nie gwarantują niezawodności
// i powinny być używane tylko jeżeli nie są stawiane wyższe wymagania na tym polu
// Program jest przystosowany do współpracy z przykładowym LoRa9x_RX

// Włączenie nagłówków bibliotek
#include <SPI.h>
#include <RH_RF95.h>

// Aktywacja
#define RFM95_CS 10
// Zerowanie
#define RFM95_RST 9
// Przerwanie, innym wejściem przerwań jest 3
#define RFM95_INT 2

// Zmienić na 434.0 lub inną częstotliwość zgodną z odbiornikiem
#define RF95_FREQ 915.0
```

```
// Obiekt sterownika modułu z wybranymi wejściami
RH_RF95 rf95(RFM95_CS, RFM95_INT);

void setup()
{
  pinMode(RFM95_RST, OUTPUT);
  digitalWrite(RFM95_RST, HIGH);

  while (!Serial);
  Serial.begin(9600);
  delay(100);

  Serial.println("Arduino LoRa TX Test!");

  // ręczne wyzerowanie
  digitalWrite(RFM95_RST, LOW);
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);

  while (!rf95.init()) {
    Serial.println("LoRa radio init failed");
    while (1);
  }
  Serial.println("LoRa radio init OK!");

  // Domyślnie po uruchomieniu 434.0 MHz, modulacja GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println("setFrequency failed");
    while (1);
  }
  Serial.print("Set Freq to: "); Serial.println(RF95_FREQ);

  // Domyślnie po uruchomieniu 434.0 MHz, 13dBm, Bw = 125 kHz, Cr = 4/5, Sf = 128
skoków/symbol, CRC włączona

  // Domyślna moc nadajnika 13 dBm, przy użyciu PA_BOOST.
  // Przy zastosowaniu modułów RFM95/96/97/98 z wyjściem PA_BOOST
  // zakres mocy wyjściowych rozciąga się od 5 do 23 dBm:
  rf95.setTxPower(23, false);
}

int16_t packetnum = 0;  // licznik pakietów podwyższany po każdej transmisji

void loop()
{
  Serial.println("Sending to rf95_server"); // Nadanie komunikatu do rf95_server

  char radiopacket[20] = "Hello World #      ";
  itoa(packetnum++, radiopacket+13, 10);
  Serial.print("Sending ");
  Serial.println(radiopacket);
  radiopacket[19] = 0;
```

```
  Serial.println("Sending..."); delay(10);
  rf95.send((uint8_t *)radiopacket, 20);

  Serial.println("Waiting for packet to complete..."); delay(10);
  rf95.waitPacketSent();  // Oczekiwanie na odpowiedź
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);

  Serial.println("Waiting for reply..."); delay(10);
  if (rf95.waitAvailableTimeout(1000))
  {
    // Jeżeli odpowiedź nadeszła
    if (rf95.recv(buf, &len))
   {
      Serial.print("Got reply: ");
      Serial.println((char*)buf);
      Serial.print("RSSI: ");
      Serial.println(rf95.lastRssi(), DEC);
    }
    else
    {
      Serial.println("Receive failed");
    }
  }
  else
  {
    Serial.println("No reply, is there a listener around?");
  }
  delay(1000);
}
```

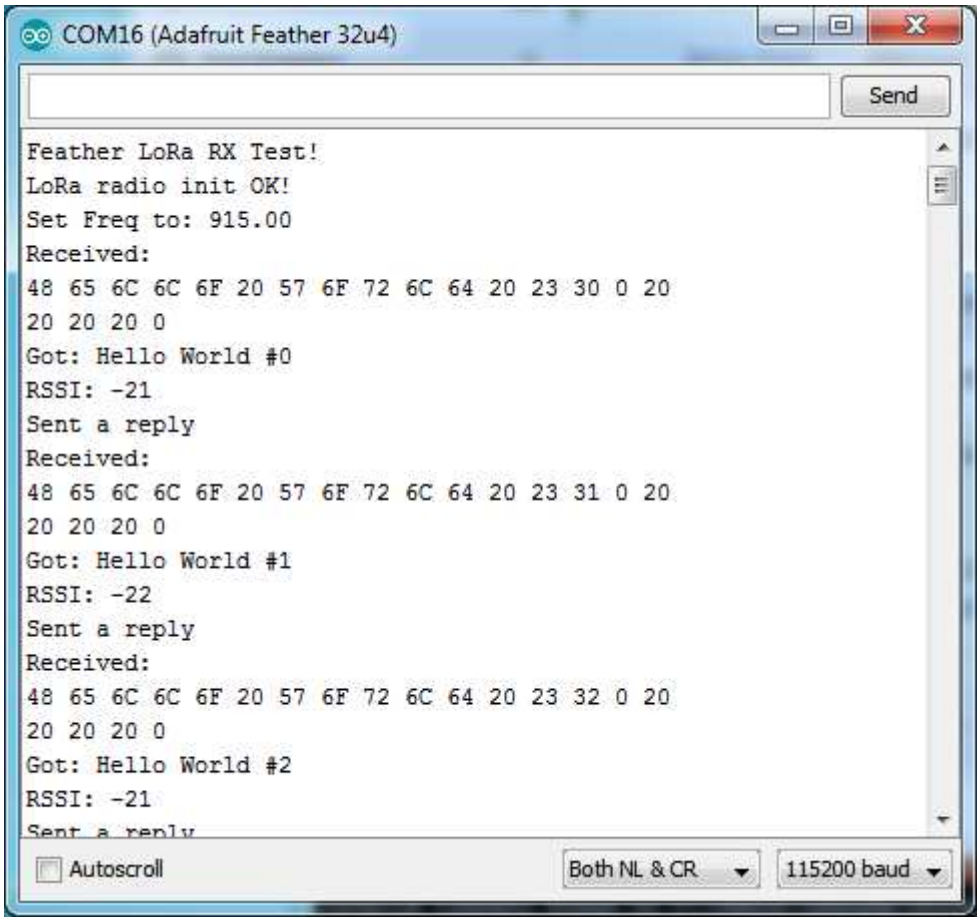


Rys. 5.2.5. Po uruchomieniu programu w oknie konsoli środowiska programistycznego „Arduino” wyświetlane są komunikaty informujace o przebiegu transmisji i zawartości nadawanych pakietów

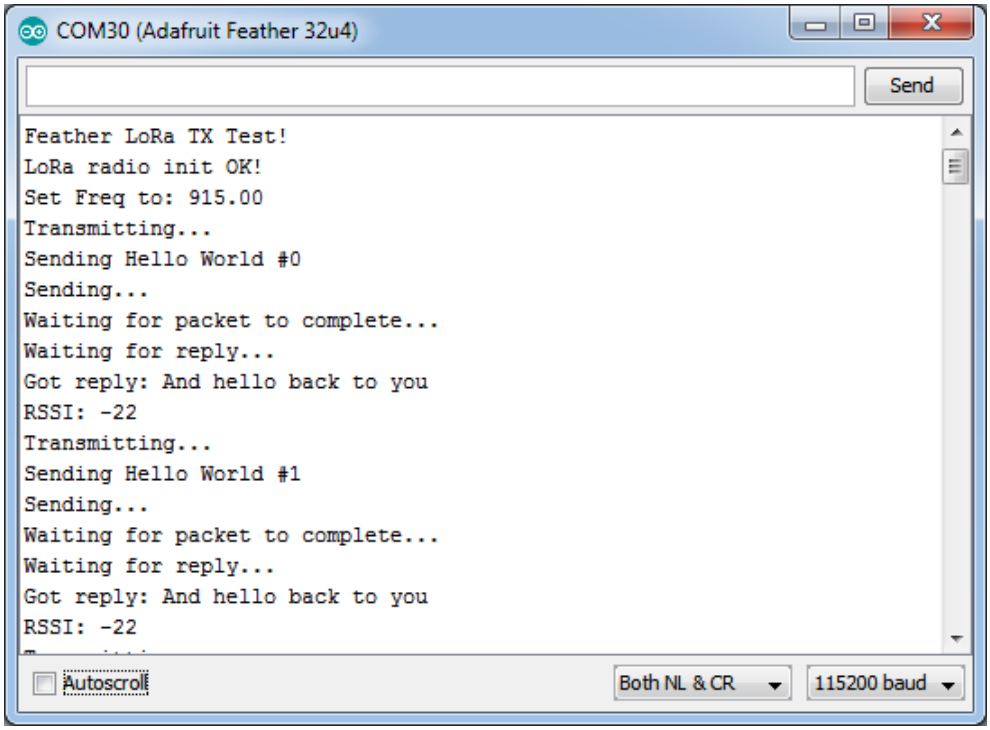
Przykładowy program odbiorczy dla „Arduino” z modułem „Dragino” i użyciem biblioteki „RadioHead” (źródło: Adafruit). Inisjalizacja modułu odbiorczego jest identyczna jak w poprzednim programie i różni się on od niego jedynie główną pętlą. Zamiast ciągłego nadawania program oczekuje na odbiór bezbłędnego komunikatu (do sprawdzenia bezbłędności służy suma kontrolna CRC) po czym wyświetla go w postaci szesnastkowej i w postaci tekstu. Dodatkow wyświetlana jest siła odebranego sygnału (RSSI). Znajduje się ona w zakresie od około -15 do -100. Wyższy numer (najwyższym jest -15) oznacza silniejszy sygnał. Na zakończenie program nadaje potwierdzenie dla nadawcy komunikatu.

```
// Arduino9x_RX
// -*- mode: C++ -*-
// Przykładowy program ilustrujący pracę prostego systemu nadawczego z modułami RH_RF95
// RH_RF95 nie posiadają możliwości adresowania i nie gwarantują niezawodności
// i powinny być używane tylko jeżeli nie są stawiane wyższe wymagania na tym polu
// Program jest przystosowany do współpracy z przykładowym Arduino9x_TX

// Włączenie nagłówków bibliotek
#include <SPI.h>
#include <RH_RF95.h>

// Aktywacja
#define RFM95_CS 10
// Zerowanie
#define RFM95_RST 9
// Przerwanie, innym wejściem przerwań jest 3
#define RFM95_INT 2

// Zmiana na  434.0 lub inną częstotliwość zgodną z używaną przez nadajnik!
#define RF95_FREQ 915.0

// Obiekt sterownika modułu z wybranymi wejściami
RH_RF95 rf95(RFM95_CS, RFM95_INT);

// Sygnalizator odbioru
#define LED 13

void setup()
{
  pinMode(LED, OUTPUT);
  pinMode(RFM95_RST, OUTPUT);
  digitalWrite(RFM95_RST, HIGH);

  while (!Serial);
  Serial.begin(9600);
  delay(100);

  Serial.println("Arduino LoRa RX Test!");

  // reżczne wyzerowanie
  digitalWrite(RFM95_RST, LOW);
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);

  while (!rf95.init()) {
```

```
    Serial.println("LoRa radio init failed");
    while (1);
  }
  Serial.println("LoRa radio init OK!");

  // Domyślnie 434.0 MHz, modulacja GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println("setFrequency failed");
    while (1);
  }
  Serial.print("Set Freq to: ");
  Serial.println(RF95_FREQ);

  // Domyślnie 434.0 MHz, 13dBm, Bw = 125 kHz, Cr = 4/5, Sf = 128 skoków/symbol, CRC włączona

  // Domyślnie moc nadajnika 13 dBm, przy użyciu PA_BOOST.
  // Przy użyciu modułów RFM95/96/97/98 z wyjściem PA_BOOST
  // zakres mocy rozciąga się od 5 do 23 dBm:
  rf95.setTxPower(23, false);
}

void loop()
{
  if (rf95.available())
  {
    // Oczekiwanie na wiadomość
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);

    if (rf95.recv(buf, &len))
    {
      digitalWrite(LED, HIGH);
      RH_RF95::printBuffer("Received: ", buf, len);
      Serial.print("Got: ");
      Serial.println((char*)buf);
       Serial.print("RSSI: ");
      Serial.println(rf95.lastRssi(), DEC);

      // Nadanie odpowiedzi
      uint8_t data[] = "And hello back to you";
      rf95.send(data, sizeof(data));
      rf95.waitPacketSent();
      Serial.println("Sent a reply");
      digitalWrite(LED, LOW);
    }
    else
    {
      Serial.println("Receive failed");
    }
  }
}
```



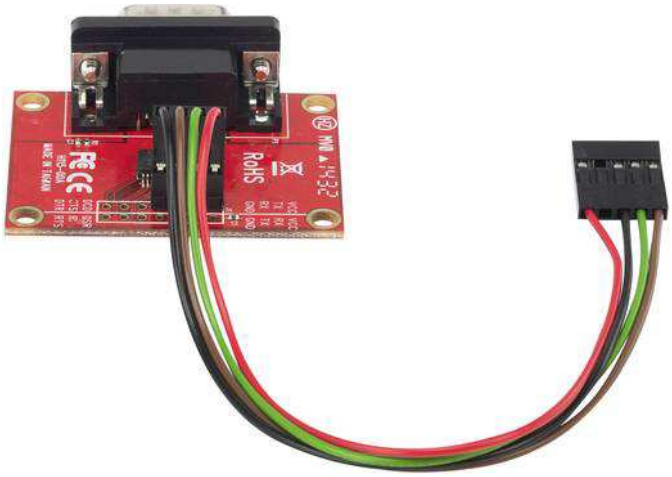
Rys. 5.2.6. Dane wyświetlane w oknie konsoli programu odbiorczego



Rys. 5.2.7. Dane wyświetlane w oknie konsoli programu nadawczego z odpowiedziami odbiornika

Złącze RS-232 dla „Maliny”

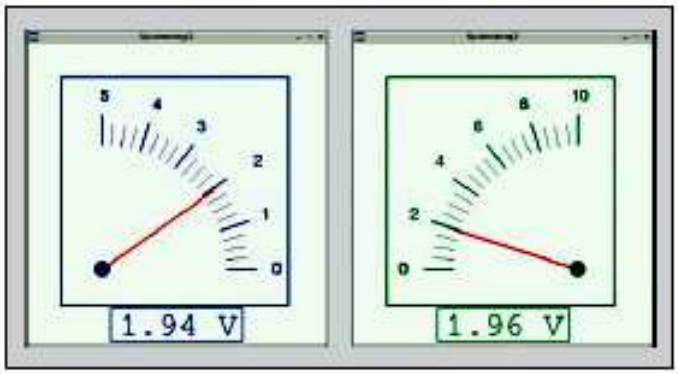
Moduł rozszerzeń ze złączem RS-232 przydaje się do połączenia „Maliny” z modemami TNC lub z innymi urządzeniami peryferyjnymi, takimi jak niektóre rodzaje czujników albo układów wykonawczych.



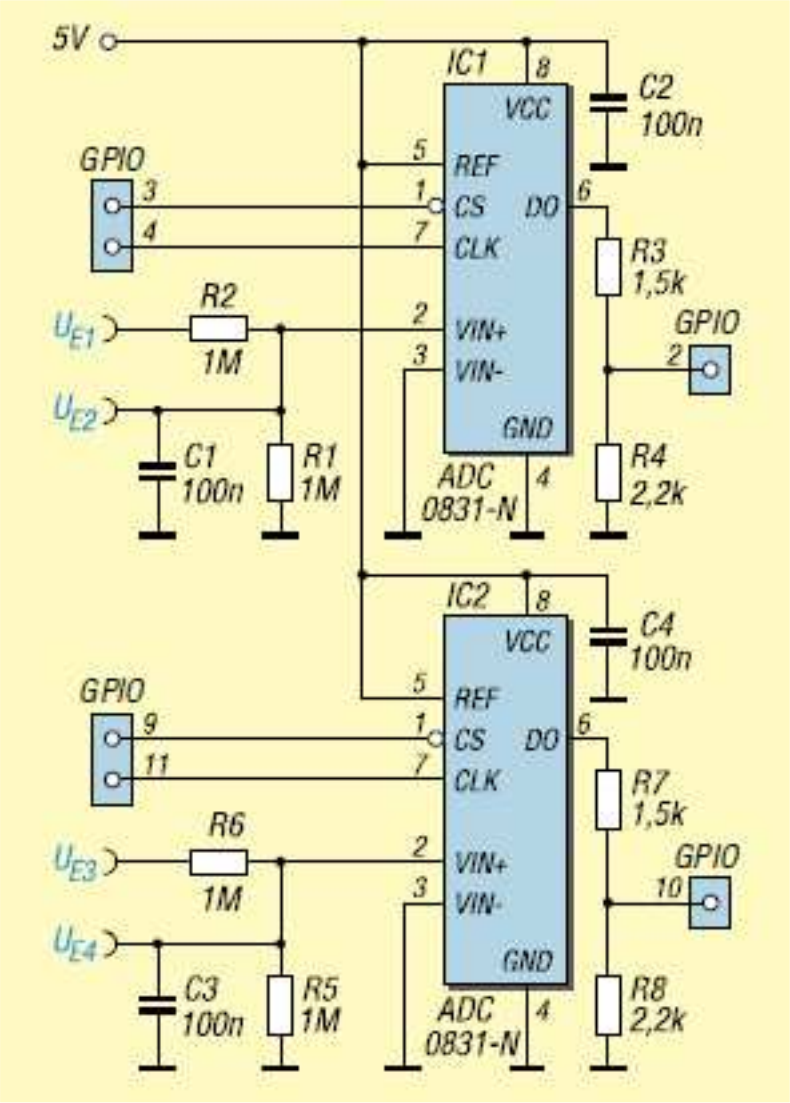
Fot. 5.3.1. Moduł złącza RS232 dla wszystkich modeli „Maliny”. Kabel służy do połączenia ze złączem GPIO

Przetworniki analogowo-cyfrowe dla „Maliny”

W odróżnieniu od „Arduino” „Malina” nie posiada własnych przetworników analogowo-cyfrowych i wymaga ich podłączenia dodatkowo. Schemat ideowy układu z dwoma przetwornikami analogowo-cyfrowymi typu ADC0831-N przedstawia rys. 5.4.2 (zamiast nich można użyć podwójnego przetwornika ADC0832). Dla zaprogramowania programu korzystającego z przetworników konieczne jest zainstalowanie biblioteki GPIO dla języka Python, za pomocą następujących poleceń wpisanych w oknie LXTerminala:
sudo apt-get update
sudo apt-get install python-rpi.gpio
Przetworniki są zasilane napieciem 5 V i konieczne jest ograniczenie napięć wejściowych do tej wartości. Wejścia E2 i E4 są przewidziane do pomiaru napięć w zakresie 5 V, natomiast wejścia E1 i E3 – w zakresie 10 V. Dla otrzymania innych zakresów pomiarowych należy odpowiednio dopasować dzielniki oporowe R2R1 i R6R5. Czas przetwarzania zależy od częstotliwości sygnału zegarowego podawanego na nóżki 7 przetworników. Zgodnie z danymi katalogowymi maksymalna częstotliwość zegarowa wynosi 400 kHz, co oznacza, że okres przemiany wynosi 32 µs. Na wejście GPIO „Maliny“ wolno podawać sygnały o napięciach nie przekraczających 3,3 V dlatego też konieczne jest zastosowanie dzielników R3R4 i R7R8.



Rys. 5.4.1. Woltomierze wyświetlane na ekranie



Rys. 5.4.2. Schemat podłączenia dwóch przetworników ADC 0831-N do złącza GPIO „Maliny”. Wejścia CS służą do włączenia wybranego przetwornika

Do wywołania przytoczonego dalej przykładowego programu odczytującego napięcia analogowe i wyświetlającego je w postaci mierników wskazówkowych na ekranie służy polecenie
sudo python Voltm01c_R.py
a do jego zakończenia kombinacja klawiszy CTRL-C w oknie terminalowym. Schemat i przykładowy program pochodzą z nru 10/2014 „Funkamateura”.

```
#!/usr/bin/env python
import time
import RPi.GPIO as GPIO
import pygame,sys
import math

#Voltm01c_mbR.py

# Wyprowadzenia GPIO połączone z ADC0831_1
AD_Dat  = 10 # przy użyciu przetwornika zasilanego Vcc= +5V
             # konieczne jest zastosowanie dzielnika oporowego
             # dla ograniczenia napięcia wyjściowego do około 3V !
             # jak to pokazano na schemacie!
```

```
AD_Clk = 11
AD_CS = 9

# Konfiguracja GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

out=GPIO.output
inp=GPIO.input
low=GPIO.LOW
hi=GPIO.HIGH

#ADC0831_1
GPIO.setup(AD_CS,GPIO.OUT)
GPIO.setup(AD_Dat,GPIO.IN)
GPIO.setup(AD_Clk,GPIO.OUT)

#Stany początkowe
out(AD_CS,hi)

def AD_einlesen():#ADC0831

   out(AD_Clk,low)
   AD_byte=0
   for n in range(8):  # Odczyt ADW 8 razy
      time.sleep(0.001)
      #out(AD_CS,hi)
      Bit7=0x80
      time.sleep(0.001)
      out(AD_CS,low)
      time.sleep(0.001)
      Analogwert=0
      out(AD_Clk,hi)     # por. opisy
      time.sleep(0.001)  #  w danych katalogowych ADC0831
      out(AD_Clk,low)
      time.sleep(0.001)
      out(AD_Clk,hi)
      time.sleep(0.001)
      out(AD_Clk,low)
      time.sleep(0.001)
      for z in range(8):#  Wczytanie 8 bitów do AD_Dat
         if (inp(AD_Dat)):
            Analogwert=Analogwert|Bit7
         out(AD_Clk,hi)
         time.sleep(0.002)
         out(AD_Clk,low)
         Bit7=Bit7>>1   # Przesunięcie w prawo o 1 bit
         time.sleep(0.002)
         Ergebnis=Analogwert
      out(AD_CS,hi)
      AD_byte=AD_byte+Analogwert
   AD_byte=AD_byte/8

   AD_Ergebnis=AD_byte
   return AD_Ergebnis
```

```
def Umrechnungen(ADWert):       # Wskazania woltomierza zależne od napięcia
                        # na zacisku 5V-Maliny
                        # "Kalibracja" "woltomierza"
                        # za pomocą napięcia porównawczego
                        # możliwa np. za pomocą woltomierza cyfrowego
                        # W tym celu wartość 1960 należy zmniejszać krokowo co 5 lub 10
                        # aż do zrównania się wskazań programu i woltomierza cyfrowego
                        # przeliczenie V =AD_Ergebnis*5V/255
   Spg_wert=ADWert           # 5/255 = 0.019607843
   Spg_wert=Spg_wert*1960 # np. ADWert=127 --> Spg_wert=127*1960=248920
   Spg_wert=Spg_wert/1000 # 248
   Darstellungswert=Spg_wert/2 # 124 dla pozycji wskazówki
   if mb1==2:
      Spg_wert=Spg_wert*2
   Ziffer_eins=Spg_wert/100    # 2
   rest_z=Spg_wert % 100       # 48
   Ziffer_zwei=rest_z/10          # 4
   Ziffer_drei=rest_z%10          # 8
   return Ziffer_eins, Ziffer_zwei, Ziffer_drei, Darstellungswert,Spg_wert


# Definicje kolorów, mogą być rozszerzone o dalsze

schwarz=(  0,   0,   0)
weiss = (255, 255, 255)
blau =  (  0,   0, 255)
h_blau=(192,255,255)
gruen = (  0, 255,   0)
m_gruen=(0,192,0)
d_gruen=(0,128,0)
rot =   (255,   0,   0)
d_rot=(192,0,0)
gelb=(255,255,0)
d_gelb=(192,192,0)
magenta=(255,0,255)
cyan=(0,255,255)
orange=(255,128,64)
d_orange=(192,64,0)
braun=(128,64,0)
lila=(128,0,255)
grau=(192,192,192)
d_grau=(128,128,128)

def Beschriftung5V():
   #Napisy "0, 1, 2, 3, 4, 5" na skali symulowanego miernika
   #Schriftart=pygame.font.Font.set_bold  #fett
   Schriftart=pygame.font.SysFont("ARIAL",int(20*dis_f))
   Label_w=Schriftart.render("0",1,VordGr_F)
   Bereich.blit(Label_w,(int((55)*dis_f),int(258*dis_f)))
   Label_w=Schriftart.render("1",1,VordGr_F)
   Bereich.blit(Label_w,(int((68)*dis_f),int(193*dis_f)))
   Label_w=Schriftart.render("2",1,VordGr_F)
   Bereich.blit(Label_w,(int((98)*dis_f),int(129*dis_f)))
   Label_w=Schriftart.render("3",1,VordGr_F)
```

```
    Bereich.blit(Label_w,(int((145)*dis_f),int(87*dis_f)))
    Label_w=Schriftart.render("4",1,VordGr_F)
    Bereich.blit(Label_w,(int((200)*dis_f),int(61*dis_f)))
    Label_w=Schriftart.render(" 5",1,VordGr_F)
    Bereich.blit(Label_w,(int((262)*dis_f),int(50*dis_f)))

def Beschriftung10V():
    #Opisy "0, 2, 4, 6, 8, 10" skali symulowanego miernika
    #Schriftart=pygame.font.Font.set_bold  #fett
    Schriftart=pygame.font.SysFont("ARIAL",int(20*dis_f))
    Label_w=Schriftart.render("0",1,VordGr_F)
    Bereich.blit(Label_w,(int((55)*dis_f),int(258*dis_f)))
    Label_w=Schriftart.render("2",1,VordGr_F)
    Bereich.blit(Label_w,(int((68)*dis_f),int(193*dis_f)))
    Label_w=Schriftart.render("4",1,VordGr_F)
    Bereich.blit(Label_w,(int((98)*dis_f),int(129*dis_f)))
    Label_w=Schriftart.render("6",1,VordGr_F)
    Bereich.blit(Label_w,(int((145)*dis_f),int(87*dis_f)))
    Label_w=Schriftart.render("8",1,VordGr_F)
    Bereich.blit(Label_w,(int((200)*dis_f),int(61*dis_f)))
    Label_w=Schriftart.render("10",1,VordGr_F)
    Bereich.blit(Label_w,(int((262)*dis_f),int(50*dis_f)))


def Messwerteinlesen():
    ADWert=AD_einlesen()

    Spg_wert_tmp=ADWert

    #Przeliczenie (ADWert)
    Ziffer_eins,Ziffer_zwei,Ziffer_drei,Darstellungswert,Spg_wert=Umrechnungen(ADWert)

    # Skasowanie obszaru wskazań

    pygame.draw.rect(Bereich,HintGr_F,(int(100*dis_f),int(317*dis_f),int(160*dis_f),int(40*dis_f)))
    pygame.draw.rect(Bereich,VordGr_F,(int(100*dis_f),int(317*dis_f),int(160*dis_f),int(40*dis_f)),2)
    #wyświetlenie bieżącej wartości
    Schriftart=pygame.font.Font.set_bold
    Schriftart=pygame.font.SysFont("COURIERNew",int(40*dis_f))
    #Dane numeryczne

Label=Schriftart.render(str(Ziffer_eins)+"."+str(Ziffer_zwei)+str(Ziffer_drei)+Einheitenzeichen,1,Vord
Gr_F)
    #
    Bereich.blit(Label,(int(110*dis_f),int(317*dis_f)))#

    # Wyświetlanie wskazówki

    x1_zg=int(270*dis_f)  # Początek wskazówki
    y1_zg=int(270*dis_f)

    # najpierw kasowanie zawartości obszaru
    pygame.draw.rect(Bereich,HintGr_F,(int(70*dis_f),int(90*dis_f),int(210*dis_f),int(200*dis_f)))

    Cosinus=math.cos(math.pi*Darstellungswert/int(1.38*360))
```

```
  Sinus=math.sin(math.pi*Darstellungswert/int(1.38*360))

  L_x=int(165*dis_f*Cosinus)#
  L_y=int(165*dis_f*Sinus)#

  if mb1==1:
     Beschriftung5V()
  if mb1==2:
     Beschriftung10V()

  # Wykreślenie nowej linii wskazówki   początek        koniec

  pygame.draw.line(Bereich,ptr_col,(x1_zg,y1_zg),(x1_zg-L_x,y1_zg-L_y),3)

 #Początek wskazówki
  pygame.draw.circle(Bereich,VordGr_F,(int(270*dis_f),int(270*dis_f)),int(dis_f*10))
 #wyświetlenie obramowania i miernika
  pygame.draw.rect(Bereich,VordGr_F,(int(40*dis_f),int(40*dis_f),int(275*dis_f),int(275*dis_f)),3)

  delta_y=20*math.sin(30*math.pi/360)
  delta_x=20*math.cos(30*math.pi/360)


  # Wykreślenie skali
  # najpierw krótkie kreski
  for k in range(21):
     Cosinus_s=math.cos(9*k*math.pi/360)#+math.pi*27.5/180)
  # punkt na łuku o promieniu 175
     L_x_s=int(175*dis_f*Cosinus_s)
     Sinus_s=math.sin(9*k*math.pi/360)#+math.pi*27.5/180)
     L_y_s=int(175*dis_f*Sinus_s)
     Cosinus=math.cos(9*k*math.pi/360)#+math.pi*27.5/180)
  # punkt na łuku o promieniu 150
     L_x=int(150*dis_f*Cosinus)
     Sinus=math.sin(9*k*math.pi/360)#+math.pi*27.5/180)
     L_y=int(150*dis_f*Sinus)
     # najpierw kasowanie czyli wykreślenie linii o kolorze tła
     pygame.draw.line(Bereich,HintGr_F,(x1_zg-L_x_s,y1_zg-L_y_s),(x1_zg-L_x,y1_zg-L_y),1)
     # teraz wykreślenie linii w kolorze pierwszoplanowym
     pygame.draw.line(Bereich,VordGr_F,(x1_zg-L_x_s,y1_zg-L_y_s),(x1_zg-L_x,y1_zg-L_y),1)
  # Wykreślenie długich kresek skali
  for k in range(6):
     Cosinus_s=math.cos(36*k*math.pi/360)
  # Punkt na łuku o promieniu 185
     L_x_s=int(185*dis_f*Cosinus_s)
     Sinus_s=math.sin(36*k*math.pi/360)
     L_y_s=int(185*dis_f*Sinus_s)
     Cosinus=math.cos(36*k*math.pi/360)
   # punkt na łuku o promieniu 150
     L_x=int(150*dis_f*Cosinus)
     Sinus=math.sin(36*k*math.pi/360)
     L_y=int(150*dis_f*Sinus)

     # linia wskazówki w kolorze pierwszoplanowym
     pygame.draw.line(Bereich,VordGr_F,(x1_zg-L_x_s,y1_zg-L_y_s),(x1_zg-L_x,y1_zg-L_y),3)
```

```
    pygame.display.update()

# Kolory i wartości do wyświetlania
print
print "*** Hintergrundfarbe ***" # tło
print"schwarz,weiss,blau,h_blau,gruen,m_gruen"
print"d_gruen,rot,d_rot,gelb,d_gelb,magenta"
print"cyan,orange,d_orange,braun,lila,grau,d_grau"
HintGr_F = input("Hintergrundfarbe:")
print
print "*** Vordergrundfarbe ***" # kolor pierwszego planu (elementów obrazu)
print"schwarz,weiss,blau,h_blau,gruen,m_gruen"
print"d_gruen,rot,d_rot,gelb,d_gelb,magenta"
print"cyan,orange,d_orange,braun,lila,grau,d_grau"
print
VordGr_F = input("Vordergrundfarbe:")
print "*** Zeigerfarbe ***" # kolor wskazówki
print"schwarz,weiss,blau,h_blau,gruen,m_gruen"
print"d_gruen,rot,d_rot,gelb,d_gelb,magenta"
print"cyan,orange,d_orange,braun,lila,grau,d_grau"
ptr_col = input("Zeigerfarbe:")
print
Caption_disp=raw_input("Bezeichnung: ")
print
Einheitenzeichen=raw_input("Einheitenzeichen: ")
print
print "Messbereich fuer U1: 5V (mb1=1) 10V (mb1=2)"
mb1=input("Faktor mb1:")
#Festlegungen: Width:500,Height:380"
dis_f=input("Darstellungsfaktor: ") # Współczynnik skalowania obrazu
print
pygame.init()
Bereich =pygame.display.set_mode((int(360*dis_f),int(360*dis_f)))
Bereich.fill(HintGr_F)
pygame.display.set_caption(Caption_disp)

Messwerteinlesen()

while True:

    Messwerteinlesen()

    GPIO.Cleanup()
```

Dodatek A

Kalibracja czujników wilgotności

W warunkach domowych lub ogólnie mówiąc amatorskich kalibrację czujników wilgotności w najprostszy sposób można przeprowadzić rozpuszczając w dostatecznie dużym i szczelnie zamykanym słoiku wybrane rodzaje soli tak, aby otrzymać ich roztwory nasycone (na dnie roztworu pozostają nierozpuszczone kryształy). Po uzyskaniu roztworu i zamknięciu słoika należy przez około pół godziny mieszać za pomocą małego wentylatorka zawarte w nim powietrze utrzymując słoik w stałej temperaturze tak, aby uzyskać stan ustalony. W tabeli A.1. podanych jest wprawdzie więcej rodzajów soli, ale w praktyce wystarczają dwa lub trzy z nich tak, aby uzyskać wilgotności względne w pobliżu górnej i dolnej granicy zakresu pomiarowego i ewentualnie w pobliżu jego środka. Dla wartości pomiędzy nimi można założyć liniowy przebieg charakterystyki czujnika.

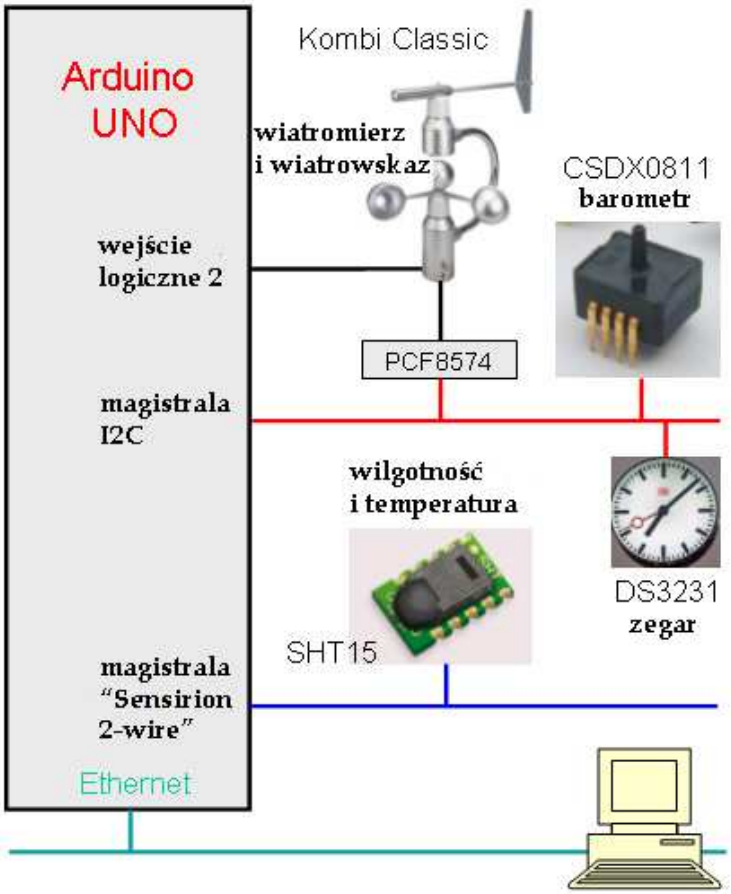
Tabela A.1. Wzorcowe wilgotności względne nasyconych roztworów soli w różnych temperaturach

Roztwór soli	**Temperatura (°C)**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
	wilgotność względna (%)								
K_2SO_4		97	97	97	96	96	96	96	96
KNO_3		94	93	92	91	89	88	85	82
KCL		87	86	85	85	84	82	81	80
$(NH_4)_2SO_4$		81	81	80	80	80	79	79	78
NaCl		76	76	75	75	75	75	75	75
$NaNO_2$		-	65	65	63	62	62	59	59
NH_4NO_3		69	65	62	59	55	53	47	42
$Na_2Cr_2O_7$		56	55	54	52	51	50	47	-
$Mg(NO_3)_2$		56	55	53	52	50	49	46	-
K_2CO_3		44	44	43	43	43	42	-	-
$MgCl_2$		34	33	33	33	32	32	31	30
CH_3COOK		21	22	22	22	21	20	-	-
LiCl		13	12	12	12	12	11	11	11

Dodatek B

Stacja meteorologiczna z serwerem HTTP na Arduino

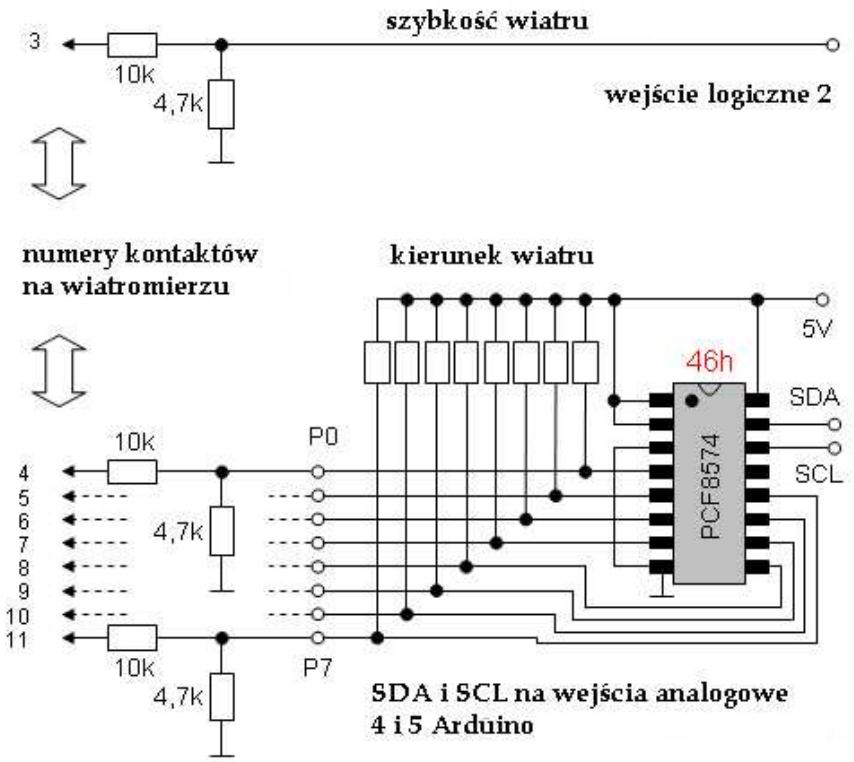
Ciekawym rozwiązaniem jest stacja meteorologiczna uruchomiona przez grupę F34 DARC. W odróżnieniu od poprzednio omówionych konstrukcji dane nie są transmitowane w postaci komunikatów drogą radiową, a są udostępniane przez pracujący na sterującym nią mikrokomputerze „Arduino UNO" serwer http. W oryginalnej konstrukcji dzięki użyciu modułu ethernetowego dla „Arduino" serwer jest dostępny w sieci lokalnej. Przy wykorzystaniu usług w rodzaju *dyn-dns* lub *no-ip* może on być też dostępny z zewnątrz przez Internet. Zamiast modułu Ethernetu można też, po uwzględnieniu tego faktu w programie zastosować moduł rozszerzeń WiFi. Alternatywnie zamiast w Internecie dane można udostępnić w Hamnecie.



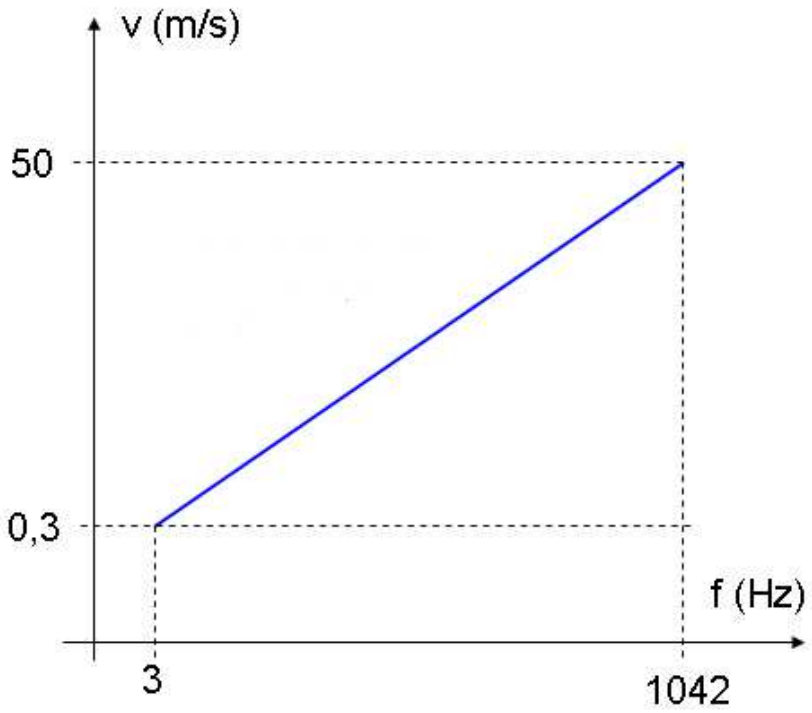
Rys. B.1. Schemat blokowy stacji meteorologicznej

W skład stacji wchodzą zespolony wiatromierz i wiatrowskaz firmy „Thies Clima" typu „Combi Classic", termometr i wilgościomierz SHT15 firmy „Sensirion" oraz barometr typu CSDX0811BARO firmy „Sensor Technics". W zależności od konkretnej sytuacji i potrzeb można ograniczyć się tylko do części z nich, rezygnując przykładowo z obserwacji wiatru.
Szybkość wiatru jest przetwarzana w czujniku na częstotliwość zależną od niej liniowo i jest ona wczytywana na wejściu logicznym 2. Sygnał pojawiający się na nim wywołuje podprogram przerwań.
Kierunek wiatru w postaci 8-bitowego słowa w kodzie Graya (co daje rozdzielczość kątową 2,5 ^{o}) jest podawana na przetwornik PCF8574 i doprowadzana do mikrokomputera przez magistralę I2C.
Temperatura i wilgoność względna są odczytywane przez „Arduino" z SHT15 również za pośrednictwem magistrali I2C, do której podłączony jest także obwód zegarowy DS3231.
Barometr CSDX0811 dostarcza danych o ciśnieniu atmosferycznym w postaci dwóch bajtów również przez magistralę I2C. Zamiast „Arduino UNO" można zastosować oczywiście „Arduino MEDA 2560".

Wiatromierz jest zasilany napięciem +15 V dlatego też jej wyjściowe sygnały logiczne muszą być podane na „Arduino” poprzez dzielniki napięcia.

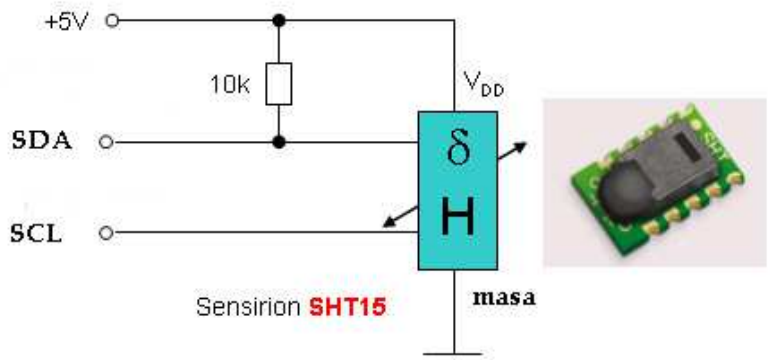


Rys. B.2. Schemat podłączenia kombinowanego wiatromierza i wiatrowskazu

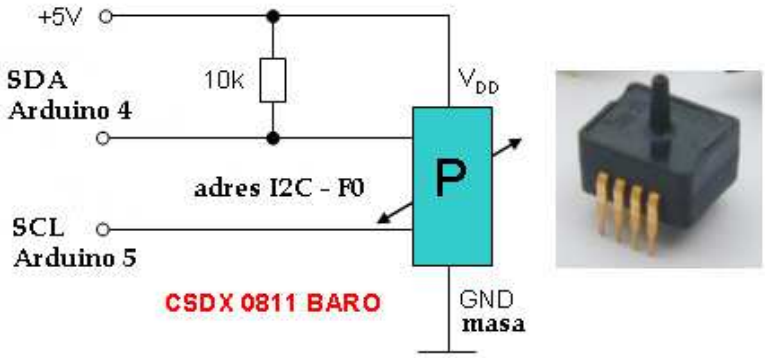


Rys. B.3. Zależność częstotliwości i mierzonej szybkości wiatru

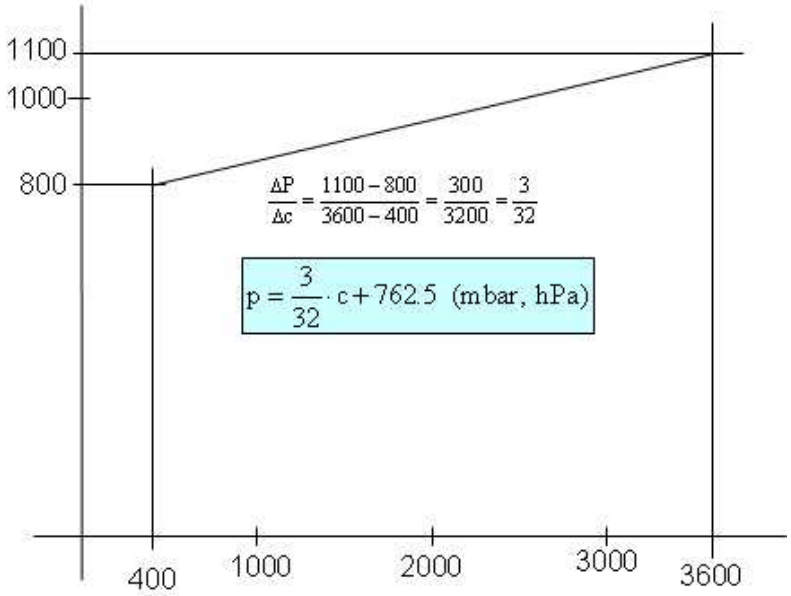
Zmierzoną szybkość wiatru oblicza się ze wzoru:
v (m/s) = 0,04783446 (m/s)/Hz * f (Hz) + 0,15649663 m/s



Rys. B.4. Schemat podłączenia termometro-wilgociomierza



Rys. B.5. Schemat podłączenia barometru



Rys. B.6. Charakterystyka barometru

Do odczytanej wartości ciśnienia należy dodać poprawkę zależną od wysokości lokalizacji stacji n.p.m. W dużym przybliżeniu można przyjąć ją jako 1 hPa na każde 8 m wysokości. Przy większych wysokościach konieczne jest uwzględnienie nieliniowego charakteru zależności.

Przykładowe oprogramowanie stacji wraz z prostym serwerem http przedstawiono poniżej.

```
// program WeatherStation_1_4.pde
// autor: Hartmut, dl8pb
// data: 04.06.2011
// usunięto zbędne zapisy w buforze przy wyświetlaniu witryny
// niezbędna dodatkowa biblioteka sensirion.h

// Zmiany
// - Włączanie i wyłączanie przerwań dla pomiaru szybkości wiatru
// przeniesione do wywołania funkcji
// - nowy sposób dynamicznego wyświetlania strony: najpierw wpisanie
// danych w jednym ciągu, a dopiero potem przekazanie funkcji

// ************************************************************************
// ************************* Opis ***************************************
/* ************************************************************************

Program odczytuje szybkość wiatru, kierunek, temperaturę, wilgotność i
ciśnienie atmosferyczne z odpowiednich czujników i udostępnia je w prostej
witrynie korzystając nieskomplikowanefo serwera http

** microkomputer arduino uno (patrz www.arduino.cc)

** wyprowadzenia arduino:
   WS5100 moduł ethernetowy / SPI: we./wy. logiczne 10, 11,12, 13
Duemilanove (wyprowadzenia 50, 51, 52, dla Arduino Mega)
   Moduł SD wyprowadzenie logiczne 4

** Czujniki
   wiatr (szybkość, kierunek):  Thies Clima, Combined Windsensor Classic
4.3324.31.000, www.thiesclima.com
   wilgotność, temperatura:    Sensirion, SHT15, www.sensirion.com
   barometr:             SensorTechnics, CSD0811BARO, www.sensortechnics.com

** wyprowadzenie magistrali i2c
   Arduino wejście analogowe 5 -> SCL
   Arduion wejście analogowe 4 -> SDA

** konfiguracja czujnika wilgotności i temperatury Sensirion SHT 15
   Arduino wyprowadzenie logiczne 3 -> takt
   Arduino wyprowadzenie logiczne 6 -> dane

** zewnętrzne przerwanie 0 dla pomiarów szybkości wiatru: wyprowadzenie 2

** licencja
   Kod dostępny publicznie na zasadach „Creative Commons Attribution-
ShareAlike 3.0 Licence”.
*/

// ************************************************************************
// ********************** włączone biblioteki *****************************
// ************************************************************************
#include <avr/pgmspace.h>
#include <Wire.h>
#include <sensirion.h>
#include <SPI.h>
#include <Ethernet.h>
#include <string.h>

// ******************* definicje ethernetowe ****************************
// Poniżej adresy MAC i  IP.
// Adres IP zależny od lokalnej sieci:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEE };
```

```
byte ip[] = {192,168,0,178};
// Inicjalizacja serwera i łącza ethernetowego przy użyciu adresu IP
// i kanału logicznego dla HTTP (standardowo 80):
Server server(80);

// licznik dla pętli loop()
int loop_count = 0;

// ************************************************************************
// ********** poniżej funkcje odczytu danych z czujników używane przez ****
// ********** stronę internetową w funkcji loop() ************************
// ************************************************************************

//------ część danych witryny powinna być umieszczona w pamięci programu
EPROM zamiast w roboczej SRAM dla uniknęcia problemów związanych z jej
ograniczoną pojemnością ------
//--niezbędne dla utworzenia strony (poniżej w sekcji loop()
// część statyczna, bufor maks. 256 bajtów!---------------
PROGMEM prog_char http_line_header[]  ="HTTP/1.0 200 OK\r\nServer:
arduino\r\nCache-Control: no-store, no-cache, must-revalidate\r\nPragma: no-
cache\r\nConnection: close\r\nContent-type: text/html; charset=utf-
8\r\n\r\n";
PROGMEM prog_char http_line_body_01[] ="<!DOCTYPE HTML PUBLIC \"-//W3C//DTD
HTML 4.01
Transitional//EN\"\"http://www.w3.org/TR/html4/loose.dtd\"><HTML><HEAD><titl
e>AVSK Wetter</title><meta http-equiv=\"refresh\" content=\" 15; URL= / \"
>";
PROGMEM prog_char http_line_body_02[] ="<style type=\"text/css\">*{font-
family:Helvetica,Arial,sans-serif;} table{border-collapse:collapse;
background-color:#d0ffff;} th{font-size:85%;} td{font-
size:80%;}</style></HEAD><BODY>";
PROGMEM prog_char http_line_body_03[] ="<table><tr><th colspan=\"2\"
align=\"left\">  Kn&uuml;ll - Wetter</th></tr>";
PROGMEM prog_char http_line_body_04[] ="<tr><td
width=\"105\">  Temperatur: <b></td><td width=\"105\">";
PROGMEM prog_char http_line_body_05[]
="</td></tr><tr><td>  Luftfeuchte:<b></td><td>";
PROGMEM prog_char http_line_body_06[]
="</td></tr><tr><td>  Taupunkt: <b></td><td>";
PROGMEM prog_char http_line_body_07[] ="</td></tr><tr><td>  rel.
Luftdruck: <b></td><td>";
PROGMEM prog_char http_line_body_08[]
="</td></tr><tr><td>  Windrichtung: <b></td><td>";
PROGMEM prog_char http_line_body_09[]
="</td></tr><tr><td>  Windgeschw.: <b></td><td>";
PROGMEM prog_char http_line_body_10[]
="</td></tr><tr><td>  CET:</td><td>";
PROGMEM prog_char http_line_tail[]    ="</td></tr></table></body> </HTML>";
PGM_P http_line[] PROGMEM = {http_line_header, http_line_body_01,
http_line_body_02,
http_line_body_03,http_line_body_04,http_line_body_05,http_line_body_06,http
_line_body_07,http_line_body_08,http_line_body_09,http_line_body_10,http_lin
e_tail}; //
#define HTTP_HEADER 0
#define HTTP_BODY   1
#define HTML_HEAD   2
#define HTML_BODY   3
#define TEMPERAT    4
#define HUMIDITY    5
#define DEW_POINT   6
#define AIR_PRESS   7
#define W_DIRECT    8
```

```
#define W_SPEED     9
#define TIME_VAL   10
#define HTML_TAIL  11

#define BUFFER_SIZE 256
#define TEMP_BUFFER_SIZE 4
char buffer[BUFFER_SIZE]; // bufor w SRAM dla danych HTTP znajdujących się w
pamięci programu
char temp_buf[4];

// funkcja poniżej wyświetla dane http z pamięci programu
void funcContentPrinter(Client client, int array_position){
     strcpy_P(buffer,(char*)pgm_read_word(&(http_line[array_position])));
     client.print(buffer);
     memset(buffer,0,BUFFER_SIZE); // kasowanie zawartości bufora
}

// ********************* odczyt kierunku wiatru *************************
// tabela konwersji kodu Graya na dziesiętny (Thiess Kombigeber,
Windrichtung). Tabela powinna być umieszczona w pamięci programu
// typy danych zdefiniowane w pgmspace.h
PROGMEM prog_char arrGrayToDezimal[] = {
0,1,3,2,7,6,4,5,15,14,12,13,8,9,11,10,31,30,28,29,24,25,27,26,16,17,19,18,23
,22,20,21,63,62,
60,61,56,57,59,58,48,49,51,50,55,54,52,53,32,33,35,34,39,38,36,37,47,46,44,4
5,40,41,43,42,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,64,65,67,66,71,70,68,69,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,143,142,140,141,136,137,139,
138,128,129,131,
130,135,134,132,133,112,113,115,114,119,118,116,117,127,126,124,125,120,121,
123,122,80,81,83,
82,87,86,84,85,95,94,92,93,88,89,91,90,111,110,108,109,104,105,107,106,96,97
,99,98,103,102,100,
101,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,79,78,76
,77,72,73,75,74, };

float degree;
float funcWindrichtung(void){
  // odczyt kierunku w kodzie Graya przez magistralę i2c (pcf8574A)
  Wire.beginTransmission(0x38); // adres PCF8574
  Wire.requestFrom(0x38, 1); // zapytanie i2c
  int gray_code_val = Wire.receive();  // wartość w kodzie Graya z
zespolonego czujnika
  Wire.endTransmission();
  int gray2decimal = pgm_read_byte_near(arrGrayToDezimal + gray_code_val);
  // pobranie wartość z tabeli konwersji do pamięci roboczej SRAM
  degree=2.5*gray2decimal;
  return(degree);
} // kierunek wiatru

char* strWindrichtung[]=
{"Nord","N/NO","NO","O/NO","Ost","O/SO","SO","S/SO","S&uuml;d","S/SW","SW","
W/SW","West","W/NW","NW","N/NW"};
byte funcWindrichtungString(float wd) {
    byte strNum;
    if (wd > 348.75 && wd <= 360.00) strNum = 0;
    if (wd >   0.00 && wd <=  11.25) strNum = 0;
    if (wd >  11.25 && wd <=  33.75) strNum = 1;
    if (wd >  33.75 && wd <=  56.25) strNum = 2;
    if (wd >  56.25 && wd <=  78.25) strNum = 3;
    if (wd >  78.25 && wd <= 101.25) strNum = 4;
```

```
    if (wd > 101.25 && wd <= 123.75) strNum = 5;
    if (wd > 123.75 && wd <= 146.25) strNum = 6;
    if (wd > 146.25 && wd <= 168.75) strNum = 7;
    if (wd > 168.75 && wd <= 191.25) strNum = 8;
    if (wd > 191.25 && wd <= 213.75) strNum = 9;
    if (wd > 213.75 && wd <= 236.25) strNum = 10;
    if (wd > 236.25 && wd <= 258.75) strNum = 11;
    if (wd > 258.75 && wd <= 281.75) strNum = 12;
    if (wd > 281.75 && wd <= 303.75) strNum = 13;
    if (wd > 303.75 && wd <= 326.25) strNum = 14;
    if (wd > 326.25 && wd <= 348.75) strNum = 15;
    return(strNum);
}

/******************** odczyt szybkości wiatru ************************/
// funkcja przerwania; impulsy z miernika szybkości powodują przerwania
// funkcja przerwania zlicza ich liczbę
volatile int n,m; // licznik dla szybkości wiatru
void isr(){
  n++;
}

// pomiar szybkości wiatru
float wspeed;
float funcWindspeed(void){
    // Zliczanie impulsów w czasie 1 sekundy. Każdy z nich wywołuje
    // funkcję przerwania, liczba zliczana w zmiennej n;
    // zliczanie przerwań; m zawiera liczbę impulsów zliczanych w czasie l s
    n=0;
    delay(1000); // czas pomiaru – 1 s
    m=n;
    wspeed = 0.04783446 * m + 0.15649663;   // wzór do obliczania szybkości
pochodzi z danych katalogowych
    if (wspeed < 0.16)
       wspeed = 0;
    Serial.print("windspeed: "); Serial.println(wspeed);
    return(wspeed);
}

/***************  odczyt barometru ************************************/
int CSD0811BARO = 0x78; // 7-bitowy adres CSD0811BARO
int pressure_counts;
float pressure_mbar;

float funcBarometer(void){
   Wire.beginTransmission(CSD0811BARO);
    Wire.requestFrom(CSD0811BARO,2); // 2 bajty
    if (Wire.available()) {
       pressure_counts = Wire.receive() << 8;
       pressure_counts |= Wire.receive();
    }
    Wire.endTransmission();    // koniec transmisji
    pressure_mbar = 0.09375 * (float)pressure_counts + 762.5 + 73.3;
    if (pressure_mbar <= (762.5 + 73.3)){
        pressure_mbar = 0;
    }
    return (pressure_mbar);
}

/********  odczyt wilgotności i temperatury z SHT15 ******************/
// konfiguracja czujnika SHT15
const uint8_t dataPin  =  6; // wyprowadzenie danych z ‘Sensirion SHT15’
```

```
const uint8_t clockPin =  3; // wyprowadzenie taktu Sensirion SHT15
Sensirion tempSensor = Sensirion(dataPin, clockPin);

float temperature, humidity, dewpoint;
int decplace_temp, decplace_dewpoint, decplace_humidity;

void funcRead_SHT15(void){
   tempSensor.measure(&temperature, &humidity, &dewpoint);
   decplace_temp = (temperature - (int)temperature) * 10;
   decplace_dewpoint = (dewpoint - (int)dewpoint) * 10;
   decplace_humidity = (humidity - (int)humidity) * 10;
}

/******** odczyt czasu z DS3231 ***********************************/
int seconds, minutes, hours;
void funcDS3232_Read(void){
  Wire.beginTransmission(0x68); // 0x68 - adres DS3231
  Wire.send(0); // start w rejestrze, 0, automatycznie podwyższany
  Wire.endTransmission();
  Wire.requestFrom(0x68, 3); // potrzebne 3 bajty (sek, min, godz)
  while(Wire.available()){
    seconds = Wire.receive(); // pobranie sekund
    minutes = Wire.receive(); // pobranie minut
    hours = Wire.receive();   // pobranie godzin
    seconds = (((seconds & 0b11110000)>>4)*10 + (seconds & 0b00001111)); //
convert BCD to decimal
    minutes = (((minutes & 0b11110000)>>4)*10 + (minutes & 0b00001111)); //
convert BCD to decimal
    hours = (((hours & 0b00100000)>>5)*20 + ((hours & 0b00010000)>>4)*10 +
(hours & 0b00001111)); // konwersja BCD na wartość dziesiętną (w trybie 24
godzinnym)
  }// while
}

float wd;
int decplace_wd;
int decplace_wspeed;
byte strNum;

void funcReadSensorData(void){

  // ----------------  Odczyt kierunku wiatru --------------------------
  wd = funcWindrichtung();            // kierunek w stopniach
  decplace_wd = (wd - (int)wd) * 10;
  strNum = funcWindrichtungString(wd); // tekst
  // ----------------- Odczyt szybkości wiatru -------------------------
  attachInterrupt(0, isr, RISING) ; // włączenie przerwań
  wspeed = funcWindspeed();
  decplace_wspeed = (wspeed - (int)wspeed) * 10;
  detachInterrupt(0); // wyłączenie przerwania 0
  //---- Odczyt czujnika SHT15 (temperatura, wilgotność, punkt rosy) ----
  funcRead_SHT15();
  //--------------- Odczyt barometru ---------------------------------
  pressure_mbar = funcBarometer();
  // -------------- Odczyt czasu przez I2C z DS3132 --------------------
  funcDS3232_Read();

}

void funcContentPrinterDyn(Client client){
  Serial.println("funcConentPrinter()");
  // ****   utworzenie dynamicznej zawartości strony w buforze ********
```

```
// Zebranie w buforze danych z czujników dla strony HTML,
// dla ich późniejszego wydania przez serwer

memset(buffer,0,BUFFER_SIZE);  // skasowanie buforów
memset(temp_buf,0,TEMP_BUFFER_SIZE);

// wpisanie danych temperatury
strcpy_P(buffer,(char*)pgm_read_word(&(http_line[TEMPERAT])));
itoa((int)temperature, temp_buf, 10);
strcat(buffer,temp_buf);
strcat(buffer,".");
memset(temp_buf,0,TEMP_BUFFER_SIZE);
itoa(abs(decplace_temp), temp_buf, 10);
strcat(buffer,temp_buf);
strcat(buffer," &ordm;C");

// wpisanie danych wilgotności
memset(temp_buf,0,TEMP_BUFFER_SIZE);
strcat_P(buffer,(char*)pgm_read_word(&(http_line[HUMIDITY])));
itoa((int)humidity, temp_buf, 10);
strcat(buffer,temp_buf);
strcat(buffer,".");
memset(temp_buf,0,TEMP_BUFFER_SIZE);
itoa(abs(decplace_humidity), temp_buf, 10);
strcat(buffer,temp_buf);
strcat(buffer," % ");

// przekazanie danych do funkcji TCPIP
client.print(buffer);
memset(buffer,0,BUFFER_SIZE); // skasowanie zawartości

// wpisanie punktu rosy
memset(temp_buf,0,TEMP_BUFFER_SIZE);
strcpy_P(buffer,(char*)pgm_read_word(&(http_line[DEW_POINT])));
itoa((int)dewpoint, temp_buf, 10);
strcat(buffer,temp_buf);
strcat(buffer,".");
memset(temp_buf,0,TEMP_BUFFER_SIZE);
itoa(abs(decplace_dewpoint), temp_buf, 10);
strcat(buffer,temp_buf);
strcat(buffer," &ordm;C");

// wpisanie danych ciśnienia
memset(temp_buf,0,TEMP_BUFFER_SIZE);
strcat_P(buffer,(char*)pgm_read_word(&(http_line[AIR_PRESS])));
itoa(pressure_mbar, temp_buf, 10);
strcat(buffer,temp_buf);
strcat(buffer,"  hPa ");

// przekazanie danych z bufora dynamicznego do funkcji TCPIP
client.print(buffer);
memset(buffer,0,BUFFER_SIZE); // skasowanie danych z bufora

// wpisanie danych kierunku wiatru
memset(temp_buf,0,TEMP_BUFFER_SIZE);
strcpy_P(buffer,(char*)pgm_read_word(&(http_line[W_DIRECT])));
itoa((int)wd, temp_buf, 10);           // całkowita część wd
strcat(buffer,temp_buf);
strcat(buffer,".");
memset(temp_buf,0,TEMP_BUFFER_SIZE);
itoa(decplace_wd, temp_buf, 10);  // ułamkowa część wd
strcat(buffer,temp_buf);
```

```
  strcat(buffer," &ordm;, ");
  strcat(buffer,strWindrichtung[strNum]);

  // wpisanie danych o szybkości wiatru
  memset(temp_buf,0,TEMP_BUFFER_SIZE);
  strcat_P(buffer,(char*)pgm_read_word(&(http_line[W_SPEED])));
  itoa((int)wspeed, temp_buf, 10);           // część całkowita wspeed
  strcat(buffer,temp_buf);
  strcat(buffer,".");
  memset(temp_buf,0,TEMP_BUFFER_SIZE);
  itoa(decplace_wspeed, temp_buf, 10);  // część ułamkowa wspeed
  strcat(buffer,temp_buf);
  strcat(buffer," m/s");

  // przekazanie danych z bufora dynamicznego do funkcji TCPIP
  client.print(buffer);
  memset(buffer,0,BUFFER_SIZE); // skasowanie zawartości bufora

  // czas
  strcpy_P(buffer,(char*)pgm_read_word(&(http_line[TIME_VAL])));
  // godziny
  memset(temp_buf,0,TEMP_BUFFER_SIZE);
  if (hours < 10)
    strcat(buffer,"0");
  itoa(hours, temp_buf, 10);
  strcat(buffer,temp_buf);
  strcat(buffer,":");
  //  minuty
  memset(temp_buf,0,TEMP_BUFFER_SIZE);
  if (minutes < 10)
     strcat(buffer,"0");
  itoa(minutes, temp_buf, 10);
  strcat(buffer,temp_buf);
  strcat(buffer,":");
  // sekundy
  memset(temp_buf,0,TEMP_BUFFER_SIZE);
  if (seconds < 10)
    strcat(buffer,"0");
  itoa(seconds, temp_buf, 10);
  strcat(buffer,temp_buf);
  memset(temp_buf,0,TEMP_BUFFER_SIZE);

  // przekazanie danych z bufora dynamicznego do funkcji TCPIP
  client.print(buffer);
  memset(buffer,0,BUFFER_SIZE); // zerowanie zawartości bufora
}

// ************************************************************************
// ********************** sekcja aduino setup() ***************************
// ************************************************************************
void setup() {
  Serial.begin(115200);

  // uruchomienie połączenia ethernetowego i serwera:
  Ethernet.begin(mac, ip);
  server.begin();

  Wire.begin();
     // ustawienie wyjść PCF8574 na ‘1’ przed odczytem
     Wire.beginTransmission(0x38); // adres PCF8574
     Wire.send(255); // wydanie wszystkich jedynek
     Wire.endTransmission();
```

```
     // inicjalizacja DS3231
     // zerowanie bitu /EOSC
     // czasem może być konieczne upewnienie się, że zegar chodzi
     // przy zasilaniu bateryjnym. Normalnie po nastawieniu zerowanie
     // nie jest konieczne, ale dobrze to sprawdzić.
     // (patrz www.macetech.com)
     Wire.beginTransmission(0x68); // adres DS3231
     Wire.send(0x0E); // wybór rejestru
     Wire.send(0b00011100); // zapis bitów w rejestrze, bit 7 /EOSC
     Wire.endTransmission();
  // inicjalizacja przerwań dla pomiaru szybkości wiatru
  pinMode(2,INPUT);
  // attachInterrupt(0, isr, RISING) ; // włączenie przerwania 0
  // detachInterrupt(0); // wyłączenie przerwania 0
  Serial.println("program WeatherStation_1_4_beta.pde\nIP:
192.168.1.178\n");
}

// ************************************************************************
// ***************** sekcja arduino loop() – serwer HTTP *****************
// ************************************************************************
void loop(){

  // -- oczekiwanie na odbiór zapytań klientów i wyświetlanie danych na
witrynie --
  Client client = server.available();
    if (client) {
      Serial.println("");
      while (client.connected()) {
         if (client.available()) {
           loop_count++;  //DEBUG
           Serial.print("loop("); Serial.print(loop_count);
Serial.println(")");
           // odczyt czujników
           funcReadSensorData();
           // wyświetlanie strony
           funcContentPrinter(client,HTTP_HEADER);
           funcContentPrinter(client,HTTP_BODY  );
           funcContentPrinter(client,HTML_HEAD  );
           funcContentPrinter(client,HTML_BODY  );
           funcContentPrinterDyn(client);            // zawartość dynamiczna
           funcContentPrinter(client,HTML_TAIL  );
           break;
         } // if (client.available())
    } //while(client.conected)
    // czas dla przeglądarki na odbiór danych
    delay(1);
    // koniec połączenia:
    client.stop();
  }// if(client)

} // loop()
```

Literatura i adresy internetowe

[1] www.arduino.cc – główna witryna projektu „arduino”
[2] sklep.avt.pl – sklep internetowy wydawnictwa AVT
[3] www.swiatradio.com.pl – witryna internetowa „Świata Radio”
[4] www.kg-gps.de
[5] www.dragino.com – witryna producenta nakładek „LoRa” dla „Arduino”
[6] aprs.fi – obserwacja położenia stacji APRS i ich komunikatów
[7] www.wxnet.de – konstrukcje czujników
[8[www.umnicom.de – konstrukcje czujników
[9] www.sander-electronic.de
[10] www.elektronik-labor.de
[11] www.aatis.de
[12] www.progettiarduino.com

[110] „Arduino projects for Ham Radio”, Glen Popiel, KW5GP, wyd. ARRL, 2017
[111] www.ui-view.org/files/APRS101.pdf – definicja standardu APRS
[112] “Funkamateur”, roczniki 2006 – 2017
[113] „Praxixheft” 1– 17, wydawnictwo AATiS
[114] „Selbstgebaute Detektoren für Strahlenquellen in der Umwelt”, Peter Lay, Franzis Verlag, Poing 2011

W serii „Biblioteka polskiego krótkofalowca” dotychczas ukazały się:

Nr 1 – „Poradnik D-STAR”, wydanie 1 i 2
Nr 2 – „Instrukcja do programu D-RATS”
Nr 3 – „Technika słabych sygnałów” Tom 1
Nr 4 – „Technika słabych sygnałów” Tom 2
Nr 5 – „Łączności cyfrowe na falach krótkich” Tom 1
Nr 6 – „Łączności cyfrowe na falach krótkich” Tom 2
Nr 7 – „Packet radio”
Nr 8 – „APRS i D-PRS”
Nr 9 – „Poczta elektroniczna na falach krótkich” Tom 1
Nr 10 – „Poczta elektroniczna na falach krótkich” Tom 2
Nr 11 – „Słownik niemiecko-polski i angielsko-polski” Tom 1
Nr 12 – „Radiostacje i odbiorniki z cyfrową obróbką sygnałów” Tom 1
Nr 13 – „Radiostacje i odbiorniki z cyfrową obróbką sygnałów” Tom 2
Nr 14 – „Amatorska radioastronomia”
Nr 15 – „Transmisja danych w systemie D-STAR”
Nr 16 – „Amatorska radiometeorologia”, wydanie 1 i 2
Nr 17 – „Radiolatarnie małej mocy”
Nr 18 – „Łączności na falach długich”
Nr 19 – „Poradnik Echolinku”
Nr 20 – „Arduino w krótkofalarstwie” Tom 1
Nr 21 – „Arduino w krótkofalarstwie” Tom 2
Nr 22 – „Protokół BGP w Hamnecie”
Nr 23 – „Technika słabych sygnałów” Tom 3, wydanie 1 i 2
Nr 24 – „Raspberry Pi w krótkofalarstwie”
Nr 25 – „Najpopularniejsze pasma mikrofalowe”
Nr 26 – „Poradnik DMR” wydanie 1 i 2, nr 326 – wydanie skrócone
Nr 27 – „Poradnik Hamnetu”
Nr 28 – „Budujemy Ilera” Tom 1
Nr 29 – „Budujemy Ilera” Tom 2
Nr 30 – „Konstrukcje D-Starowe”
Nr 31 – „Radiostacje i odbiorniki z cyfrową obróbką sygnałów” Tom 3
Nr 32 – „Anteny łatwe do ukrycia”
Nr 33 – „Amatorska telemetria”