AGH
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Fizyki i Informatyki Stosowanej

Praca magisterska

Michał Kowalczyk

kierunek studiów: **Fizyka Techniczna**

System do pomiaru strumieni turbulencyjnych ciepła

Opiekun: **dr hab. inż. Mirosław Zimnoch, prof. AGH**

Kraków, listopad 2020

Oświadczenie studenta

Uprzedzony(-a) o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2018 r. poz. 1191 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. — Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

.....
(czytelny podpis)

Kraków, 4 listopada 2020

Tematyka pracy magisterskiej i praktyki dyplomowej Michała Kowalczyka, studenta drugiego roku studiów drugiego stopnia na kierunku fizyka techniczna

Temat pracy magisterskiej: **System do pomiaru strumieni turbulencyjnych ciepła**

Opiekun pracy: dr hab. inż. Mirosław Zimnoch, prof. AGH
Recenzenci pracy:

Miejsce praktyki dyplomowej: WFiIS AGH, Kraków

Program pracy magisterskiej i praktyki dyplomowej

1. Omówienie realizacji pracy magisterskiej z opiekunem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
   - zapoznanie się z ideą pomiaru strumieni turbulencyjnych ciepła,
   - zapoznanie się z platformą programistyczną Arduino,
   - przygotowanie systemu pomiarowego,
   - sporządzenie sprawozdania z praktyki.
4. Przygotowanie dokumentacji technicznej.
5. Zebranie i opracowanie wyników pomiarów.
6. Dyskusja i analiza wyników.
7. Opracowanie redakcyjne pracy.

Termin oddania w dziekanacie: 2020

....
(podpis kierownika katedry) (podpis opiekuna)

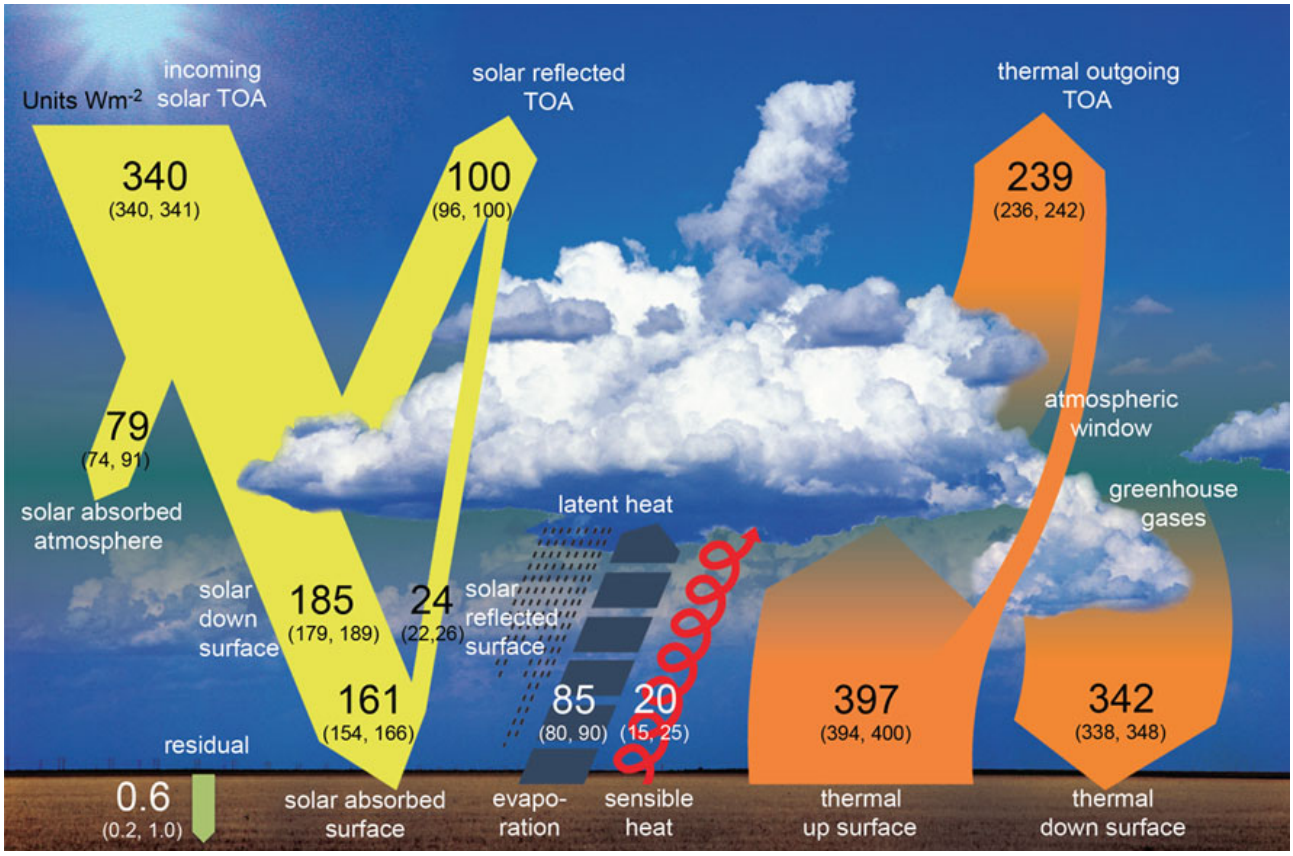
Merytoryczna ocena pracy przez opiekuna

Merytoryczna ocena pracy przez recenzenta

Spis treści

1 Wstęp

Najniższa warstwa troposfery, na którą bezpośrednio oddziałuje powierzchnia Ziemi, nazywana jest planetarną warstwą graniczną. Rozciąga się ona na wysokość od kilkudziesięciu metrów do kilku kilometrów. Warunki meteorologiczne i klimatyczne danego obszaru Ziemi w głównej mierze zależą od ilości energii słonecznej docierającej do podłoża oraz od sposobu jej kolejnych przemian w różnych procesach fizycznych i chemicznych (rysunek 1). Podstawowym kierunkiem badawczym w tej dziedzinie jest więc pomiar bilansu energetycznego warstwy granicznej, który może mieć charakter zmiany okresowej, wymuszony przez zmianę pór roku, czy przez cykl dobowy, jak i również charakter nieregularny, zależny od chwilowych sytuacji pogodowych. Jednym z istotnych składników tego bilansu są strumienie ciepła jawnego (ang. *sensible heat*) i utajonego (ang. *latent heat*).



Rysunek 1: Bilans energetyczny atmosfery ziemskiej (źródło obrazu: [37]).

Celem niniejszej pracy jest budowa i testowanie systemu pomiarowego, pozwalającego na oszacowanie wartości turbulencyjnych strumieni ciepła jawnego i utajonego w planetarnej warstwie granicznej atmosfery.

2 Wymiana energii w obszarze podłoża i planetarnej warstwy granicznej

2.1 Podstawowe zagadnienia związane z wymianą energii

Wymiana energii w otoczeniu planetarnej warstwy granicznej związana jest z występowaniem w niej strumieni energii. Strumień energii definiuje się jako ilość energii przepływającej na danym obszarze w określonym czasie [38]:

$$\Phi_E = \frac{dE}{dt} \tag{2.1}$$

gdzie: dE – elementarna ilość energii przenoszona w czasie dt. Jednostką strumienia energii jest Wat.

Po odniesieniu strumienia energii do zorientowanej przestrzeni o określonej powierzchni, otrzymuje się wielkość zwaną gęstością strumienia energii [1] [38]:

$$d\Phi_E = \boldsymbol{\rho_E} \cdot d\boldsymbol{A} \tag{2.2}$$

gdzie: $\boldsymbol{\rho_E}$ - wektor gęstości strumienia energii $[\frac{W}{m^2}]$; $d\boldsymbol{A}$ - wektor powierzchniowy. Skalarnie zależność ta przyjmuje formę:

$$\rho_E = \frac{d\Phi_E}{dA} \tag{2.3}$$

W obszarze planetarnej warstwy granicznej wymiana ciepła, które jest formą energii, może odbywać się następującymi metodami:

- poprzez przewodzenie
- poprzez promieniowanie cieplne
- poprzez konwekcję

2.2 Wymiana ciepła przez przewodzenie

Wymiana ciepła na drodze zjawiska przewodzenia występuje, gdy w danym ośrodku pojawia się gradient temperatur. Strumień ciepła opisywany jest w takim wypadku przez prawo Fouriera:

$$G = -\lambda \frac{\partial T}{\partial z} \tag{2.4}$$

[1] W literaturze terminy *strumień ciepała* i *gęstość strumienia ciepła* używane są zamiennie w kontekście wielkości $\frac{W}{m^2}$. Taka konwencja zachowana jest również w niniejszej pracy.

gdzie: λ - współczynnik przewodzenia ciepła $[\frac{J}{s \cdot m \cdot K}]$.

Równanie to jest prawdziwe tylko w wypadku przepływu ciepła w warunkach stanu ustalonego (czyli gdy profil temperatury nie zmienia się wraz z upływem czasu) i w warunkach ośrodka jednorodnego. W przypadku przepływu w stanie nieustalonym, jaki występuje na przykład w glebie, opis przepływu ciepła przedstawia równanie:

$$\rho \cdot c \cdot \frac{\partial T}{\partial t} = -\frac{\partial G}{\partial z} \tag{2.5}$$

gdzie: ρ - gęstość ośrodka, c - ciepło właściwe ośrodka. Po połączeniu równań 2.4 i 2.5 otrzymuje się następujące równanie dyfuzji [27]:

$$\frac{\partial T}{\partial t} = \frac{\lambda}{\rho \cdot c} \cdot \frac{\partial^2 T}{\partial z^2} = \kappa_T \cdot \frac{\partial^2 T}{\partial z^2} \tag{2.6}$$

gdzie: κ_T - dyfuzyjność termiczna. Czynnik $C = \rho \cdot c$ nazywany jest pojemnością cieplną. W przypadku ośrodków złożonych z kilku frakcji, pojemność cieplna jest średnią ważoną pojemności cieplnych poszczególnych składników. Wagę stanowi udział objętościowy danej frakcji.

W ciągu dnia strumień energii promieniowania słonecznego powoduje nagrzewanie się powierzchni Ziemi, co wymusza przepływ strumieni ciepła G wgłąb powierzchni. W nocy, powierzchniowa warstwa Ziemi wychładza się, co powoduje zmianę kierunku strumienia G.

2.3 Wymiana ciepła przez promieniowanie

Radiacyjna wymiana ciepła polega na przenoszeniu energii za pomocą fal elektromagnetycznych. Głównym zakresem długości fal, których dotyczy ta forma przenoszenia energii, są fale z zakresu podczerwieni. Zjawisko to jednak zachodzi dla całego spektrum. Długość fali elektromagnetycznej przy której występuje największa moc promieniowania opisuje prawo Wiena:

$$\lambda_{max} = \frac{b}{T} \tag{2.7}$$

gdzie: $b \approx 2,90 \cdot 10^{-9} m \cdot K$ - stała Wiena, T - temperatura powierzchni ciała w Kelwinach.

Gęstość strumienia energii wypromieniowanej w danej temperaturze przez ciało doskonale czarne opisuje prawo Stefana-Boltzmanna:

$$\rho_E = \sigma \cdot T^4 \tag{2.8}$$

gdzie: $\sigma \approx 5,67 \cdot 10^{-8} \frac{W}{m^2 \cdot K^4}$ - stała Stefana-Boltzmanna. Jako, że model idealnego ciała doskonale czarnego nie występuje w rzeczywistości, do opisu emisji strumieni radiacyjnych przez ciała rzeczywiste wprowadza się wielkość ϵ zwaną względną zdolnością emisyjną. Jest to stosunek

wartości strumienia wyemitowanego przez dane ciało rzeczywiste, do strumienia jaki zostałby wyemitowany przez ciało doskonale czarne w identycznej temperaturze i w postaci fali elektromagnetycznej o identycznej długości. Prawo Stefana-Boltzmanna przyjmuje wtedy postać:

$$\rho_E = \epsilon \cdot \sigma \cdot T^4 \tag{2.9}$$

W atmosferze ziemskiej wymiana energii przez promieniowanie następuje poprzez promieniowanie krótkofalowe (pochodzenia słonecznego) i długofalowe (promieniowanie powierzchni Ziemi i atmosfery). Bilans tych strumieni można przedstawić jako [28]:

$$\rho_{KL} = \rho_{K^+} - \rho_{K^-} + \rho_{L^+} - \rho_{L^-} \tag{2.10}$$

gdzie: ρ_{K^+} - strumień energii promieniowania słonecznego, ρ_{K^-} - strumień energii odbitego promieniowania słonecznego, ρ_{L^+} - strumień energii promieniowania długofalowego atmosfery, ρ_{L^-} - strumień energii promieniowania długofalowego powierzchni Ziemi.

W ciągu dnia występują zarówno strumienie długofalowe, jak i krótkofalowe. W nocy występuje tylko część długofalowa.

2.4 Konwekcyjna wymiana ciepła

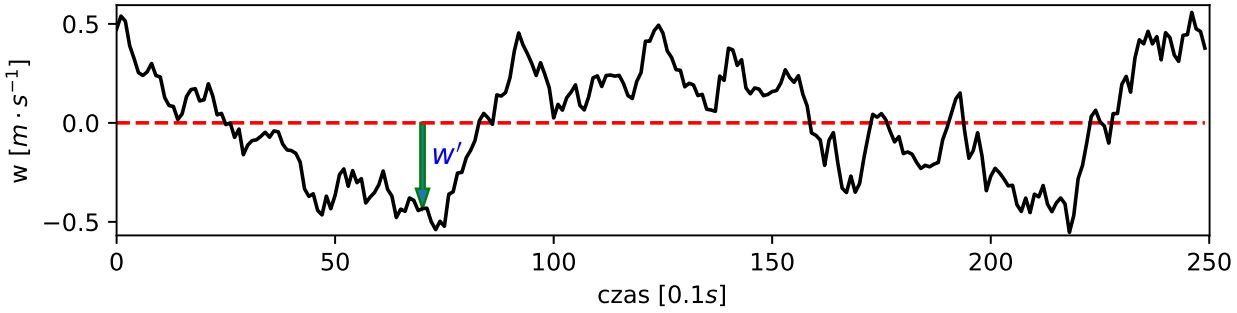
Konwekcja występuje w przypadku przemieszczania się mas ośrodka o różnych temperaturach. Jest to tak zwany transport ciepła jawnego. Może mieć ona charakter swobodny, gdy przemieszczanie występuje na wskutek działania sił powstałych w wyniku różnicy temperatur i zmiany gęstości ośrodka, lub charakter wymuszony, gdy przemieszczanie mas występuje w wyniku czynników zewnętrznych, na przykład poprzez dmuchawy lub wiatr.

Konwekcja wpływa również na przemieszczanie się pary wodnej wraz z masami powietrza. Powstają wtedy strumienie ciepła utajonego. Ciepło utajone może zostać zamienione na ciepło jawne w procesie kondensacji pary wodnej, i odwrotnie, parowanie wody może spowodować zamianę ciepła jawnego na utajone.

2.4.1 Podstawowe zagadnienia teorii turbulencji

W atmosferze ziemskiej występuje przestrzenny ruch ciepła, pędu i masy. Szczególne znaczenie, pod względem badania bilansu energetycznego atmosfery, mają transporty pionowe. Przyjmuje się, że w przypadku poziomej homogeniczności wielkości meteorologicznych, czyli w przypadku, gdy na przykład pola prędkości wiatru, temperatury i wilgotności są statystycznie

stacjonarne i jednorodne w płaszczyźnie poziomej, średnia wartość składowej pionowej prędkości wiatru $\bar{w}$ w przybliżeniu przyjmuje wartość równą zeru (co nie zawsze jest prawdą i zakładane jest jedynie w celach ułatwienia zrozumienia problemu). W takim wypadku przyjąć można, że za pionowy transport wyżej wspomnianych wielkości odpowiada wyłącznie transport turbulencyjny.



Rysunek 2: Przykładowy szereg czasowy pionowej prędkości powietrza w. Przerywana linia oznacza prędkość średnią równą 0 m/s, w' oznacza chwilową fluktuację (opracowanie własne na podstawie [18], dane wygenerowano metodą Monte Carlo).

Turbulencyjny charakter ruchu pozwala na zapisanie dowolnego parametru ξ, w określonym przedziale czasowym, jako sumę jego wartości średniej $\bar{\xi}$ i jego fluktuacji ξ' [18]. Na tej podstawie zapisać można następujące zależności:

$$\begin{aligned} u &= \bar{u} + u' \\ v &= \bar{v} + v' \\ w &= \bar{w} + w' \\ \theta &= \bar{\theta} + \theta' \\ q &= \bar{q} + q' \\ \rho &= \bar{\rho} + \rho' \end{aligned} \tag{2.11}$$

gdzie: u, v, w - składowe prędkości wiatru, $m \cdot s^{-1}$; θ - temperatura potencjalna (temperatura układu adiabatycznie sprowadzonego do ciśnienia $1000hPa$), K; q - wilgotność właściwa powietrza, $kg \cdot kg^{-1}$; ρ - gęstość powietrza, $kg \cdot m^{-3}$. Na tej podstawie pionowy chwilowy strumień turbulencyjny powietrza zapisać można jako [18]:

$$F = w \cdot \rho = \bar{w} \cdot \bar{\rho} + \bar{w} \cdot \rho' + \bar{\rho} \cdot w' + w' \cdot \rho' \tag{2.12}$$

A strumień średni po zastosowaniu uśrednienia Reynoldsa:

$$\bar{F} = \bar{w} \cdot \bar{\rho} + \bar{w} \cdot \bar{\rho'} + \bar{\rho} \cdot \bar{w'} + \overline{w' \cdot \rho'} \tag{2.13}$$

Po uwzględnieniu, że $\bar{w} = 0$, $\bar{w'} = 0$ oraz $\bar{\rho'} = 0$:

$$\bar{F} = \overline{w' \cdot \rho'} \tag{2.14}$$

Analogicznie przedstawić można pionowe strumienie turbulencyjne ciepła jawnego H i utajonego E $[W \cdot m^{-2}]$, oraz pionowy turbulencyjny strumień pędu τ $[N \cdot m^{-2}]$ [18] [28]:

$$H = c_p \cdot \rho \cdot \overline{\theta' \cdot w'} \tag{2.15}$$

$$E = l \cdot \rho \cdot \overline{q' \cdot w'} \tag{2.16}$$

$$\tau = -\rho \cdot \overline{u' \cdot w'} \tag{2.17}$$

gdzie: c_p - ciepło właściwe powietrza, l - ciepło właściwe parowania, $J \cdot kg^{-1}$. Ponadto, w celu ułatwienia zapisu, zastosowano tu konwencję pomijania wskaźnika uśredniana przy symbolach strumieni [18] [28]. W dalszej części pracy, również będzie to stosowane.

Wykorzystując pojęcie strumienia pędu, wprowadzić można wielkość zwaną prędkością tarcia u_* [28]:

$$u^* = \left(\frac{\tau}{\rho}\right)^{\frac{1}{2}} \tag{2.18}$$

Pomimo pozornej prostoty równań 2.15 - 2.17, poważny problem stanowi występujący w nich czynnik wartości średniej. Z powodu występowania w atmosferze ziemskiej zmienności dobowej, rocznej oraz występowania frontów atmosferycznych, zmian pogody, a nawet klimatu, szeregi czasowe mierzonych parametrów meteorologicznych są niestacjonarne. Nie jest więc możliwe uzyskanie średniej z próby pomiarów w identycznych warunkach pomiarowych. Zamiast tego wykonuje się uśrednianie w pewnych przedziałach czasowych. Powoduje to, że nie zawsze są spełnione warunki uśredniania Reynoldsa. Ponad to, zbiór równań opisujący przepływy turbulencyjne nie posiada rozwiązania analitycznego, gdyż liczba niewiadomych jest większa od liczby równań. Z tego powodu, rozwiązania można uzyskać stosując tylko podejście fenomenologiczne i teorię podobieństwa.

2.4.2 Teoria transportu gradientowego

Teoria transportu gradientowego polega na zastosowaniu w stosunku do równań 2.15 - 2.17 analogii do dyfuzji molekularnej. Pierwsze prawo Ficka zakłada, że gęstość strumienia przepływu danej substancji określa się jako:

$$J = -D\frac{\partial \eta}{\partial x} \tag{2.19}$$

gdzie: D - współczynnik dyfuzji, η - stężenie substancji, x - odległość. Na tej podstawie pionowe turbulencyjne strumienie ciepła jawnego, utajonego oraz pędu przyjmą postać:

$$H = c_p \cdot \rho \cdot K_H \frac{\partial \theta}{\partial z} \tag{2.20}$$

$$E = l \cdot \rho \cdot K_W \frac{\partial q}{\partial z} \tag{2.21}$$

$$\tau = -\rho \cdot K_M \frac{\partial u}{\partial z} \tag{2.22}$$

gdzie: K_H, K_W, K_M - współczynniki turbulencyjnej wymiany ciepła, pary wodnej i pędu $[m^2 \cdot s^{-1}]$. W przeciwieństwie do dyfuzji molekularnej, współczynniki te nie są stałe i zależą głównie od prędkości przepływu i współrzędnej z. Założyć można jedynie, że transport turbulencyjny pary wodnej w przyziemnej warstwie atmosfery przebiega w podobny sposób co transport ciepła. Można więc przedstawić równość: $K_W = K_H$ [28]. Teoria transportu gradientowego działa jednak dobrze tylko w przypadku występowania małych wirów turbulencyjnych [18].

2.4.3 Liczba Richardsona

W 1920 roku Lewis Fry Richardson wyznaczył kryterium pozwalające na określenie warunków powstawania i zaniku turbulencji w atmosferze. Gradientowa Liczba Richardsona, która stanowi to kryterium i uwzględnia założenia teorii transportu gradientowego, określona jest jako [28]:

$$Ri = \frac{\frac{g}{\bar{\theta}} \cdot \frac{\partial \theta}{\partial z}}{\left(\frac{\partial u}{\partial z}\right)^2} \tag{2.23}$$

gdzie: g - przyspieszenie ziemskie. Stanowi ona stosunek siły wyporu atmosfery do jej siły inercji, które mają szczególne znaczenie w genezie przepływów turbulencyjnych. Siły inercji atmosfery mogą wyłącznie powodować powstawanie turbulencji, natomiast siły wyporu mogą ją powodować lub tłumić, w zależności od tego w którą stronę działają. Wartość krytycznej liczby Richardsona przyjmuje się w zakresie $0.21 - 0.25$. Przepływ laminarny przechodzi w turbulencyjny gdy wartość Ri spadnie poniżej wartości krytycznej. W praktyce, liczbę Richardsona wyznacza się na podstawie pomiarów temperatur powietrza (którymi w przyziemnej warstwie atmosfery zastąpić można temperatury potencjalne) i prędkości wiatru na dwóch poziomach. Wzór 2.23 przyjmuje wtedy postać:

$$Ri(z_s) = \frac{g}{T_0} \cdot z_s \cdot ln\frac{z_2}{z_1} \cdot \frac{\Delta T}{(\Delta u)^2} \tag{2.24}$$

gdzie: T_0 - temperatura średnia, z_s - poziom odniesienia równy $(z_1 \cdot z_2)^{1/2}$, $\Delta T = T_2 - T_1$, $\Delta u = u_2 - u_1$.

2.4.4 Teoria podobieństwa Monina–Obuchowa

Występowanie małych wirów turbulencyjnych w atmosferze ziemskiej jest rzadko spotykane, z tego powodu teoria transportu gradientowego nie spełnia swojego zadania. Rozpoczęto więc poszukiwanie empirycznych zależności przepływów turbulencyjnych i ustalono, że w warstwie granicznej zależą one od: wysokości nad powierzchnią z, prędkości tarcia u^*, strumienia ciepła przy powierzchni ziemi H_0 i parametru wyporu $g/\bar{\theta}$. Na podstawie tych parametrów przedstawić można bezwymiarową wielkość ζ nazywaną parametrem stabilności i równą:

$$\zeta = \frac{z}{L} \tag{2.25}$$

gdzie: L - długość Obuchowa:

$$L = -\frac{(u_*)^3}{\kappa \cdot \frac{g}{\bar{\theta}} \cdot \frac{H_0}{c_p \cdot \rho}} \tag{2.26}$$

gdzie: κ - stała von Karmana (najczęściej jej przyjmowana wartość wynosi 0.4). Parametr stabilności ζ pełni podobną rolę co liczba Richardsona. W przypadku równowagi chwiejnej atmosfery przyjmuje wartości ujemne, a w przypadku równowagi stałej dodatnie. W momencie równowagi obojętnej jego wartość jest bliska lub równa zeru [18]. Zgodnie z teorią Monina–Obuchowa skonstruować można następujące charakterystyki prędkości wiatru, temperatury i wilgotności przy powierzchni ziemi:

$$u_* = \sqrt{\frac{\tau_0}{\rho}} \tag{2.27}$$

$$\theta_* = -\frac{H_0}{c_p \cdot \rho \cdot u_*} \tag{2.28}$$

$$q_* = -\frac{E_0}{l \cdot \rho \cdot u_*} \tag{2.29}$$

a na ich podstawie wprowadzić zależności:

$$\frac{\kappa \cdot z}{u_*} \cdot \frac{\partial u}{\partial z} = \phi_M(\zeta) \tag{2.30}$$

$$\frac{\kappa \cdot z}{\theta_*} \cdot \frac{\partial \theta}{\partial z} = \phi_H(\zeta) \tag{2.31}$$

$$\frac{\kappa \cdot z}{q_*} \cdot \frac{\partial q}{\partial z} = \phi_W(\zeta) \tag{2.32}$$

gdzie: $\Phi_M(\zeta)$, $\Phi_H(\zeta)$ i $\Phi_W(\zeta)$ to tzw. uniwersalne funkcje podobieństwa pionowych przepływów wiatru, temperatury i wilgotności zależne od parametru stabilności $\zeta = z/L$, które należy wyznaczyć eksperymentalnie.

Ustalono, że uniwersalne funkcje podobieństwa najczęściej przyjmują postać [18]:

$$\phi_M(\zeta) = \begin{cases} (1-\gamma_1 \cdot \zeta)^{-1/4}, & dla\ \zeta < 0 \\ 1+\beta_1 \cdot \zeta, & dla\ \zeta \geq 0 \end{cases} \tag{2.33}$$

$$\phi_H(\zeta) = \phi_W(\zeta) = \begin{cases} P_t \cdot (1-\gamma_2 \cdot \zeta)^{-1/2}, & dla\ \zeta < 0 \\ P_t, & dla\ \zeta = 0 \\ P_t \cdot (1+\beta_2 \cdot \zeta), & dla\ \zeta > 0 \end{cases} \tag{2.34}$$

gdzie, najczęściej przyjmuje się: $\gamma_1 \approx \gamma_2 \approx 15$ lub $\gamma_1 \approx \gamma_2 \approx 16$ (częściej, ze względów niepewności pomiarów przyjmuje się pierwszą wartość) i $\beta_1 \approx \beta_2 \approx 5$ [18] [28] oraz P_t w przedziale od 0.74 do 1 (w większości przypadków przyjmowane jest 1). Stosowalność powyższych współczynników, według różnych źródeł, ogranicza się do wartości parametru ζ w przedziałach od -2 do 1 [18] lub od -5 do 1 [28].

Za pomocą parametru stabilności ζ oraz funkcji podobieństwa przedstawić można również liczbę Richardsona:

$$Ri = \frac{\zeta \cdot \phi_H(\zeta)}{\phi_M^2(\zeta)} \tag{2.35}$$

Na podstawie równań 2.27 - 2.29 turbulencyjne strumienie przy powierzchni ziemi przedstawić można jako:

$$\tau_0 = \rho \cdot (u_*)^2 \tag{2.36}$$

$$H_0 = -c_p \cdot \rho \cdot u_* \cdot \theta_* \tag{2.37}$$

$$E_0 = -l \cdot \rho \cdot u_* \cdot q_* \tag{2.38}$$

2.4.5 Bulk transfer

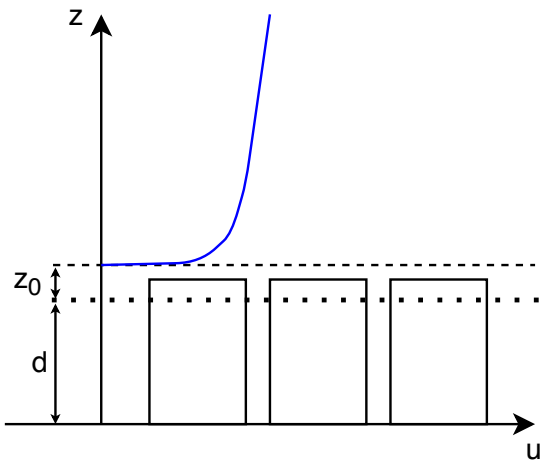
Analogicznie do równań 2.30 - 2.32, przy jednoczesnym założeniu, że w przypadku obojętnej równowagi atmosfery ($\zeta = 0$) zmiany prędkości wiatru zależą wyłącznie od wysokości z i prędkości tarcia u_*, przedstawić można bezwymiarowy profil prędkości wiatru [28]:

$$\frac{z}{u_*} \cdot \frac{\partial u}{\partial z} = const = \frac{1}{\kappa} \tag{2.39}$$

który po scałkowaniu względem z daje zależność:

$$u(z) = \frac{u_*}{\kappa} \cdot ln\frac{z}{z_0} \tag{2.40}$$

gdzie: z_0 - aerodynamiczny współczynnik szorstkości terenu. Współczynnik ten definiuje się jako wysokość, na której prędkość wiatru spada do zera i wyznacza się go eksperymentalnie.



Rysunek 3: Pionowy profil wiatru nad zwartą zabudową miejską. z_0 - parametr szorstkości terenu, d - przesunięcie płaszczyzny zerowej (opracowanie własne na podstawie [28])

Jeśli powierzchnia gruntu pokryta jest wysoką roślinnością, albo tak jak w przypadku miast zwartą zabudową, do profilu wprowadza się dodatkowy parametr d, nazywany przesunięciem płaszczyzny zerowej (rysunek 3):

$$u(z) = \frac{u_*}{\kappa} \cdot ln\frac{z-d}{z_0} \tag{2.41}$$

W celu wyznaczenia pionowych profili wiatru w przypadku stratyfikacji atmosfery innej niż obojętna, należy scałkować równanie 2.30 z funkcją ϕ_m w postaci równania 2.33. Efektem takiego działania jest następująca zależność:

$$u(z) = \frac{u_*}{\kappa} \cdot \left[ln\frac{z}{z_0} - \psi_M\left(\frac{z}{L}\right) + \psi_M\left(\frac{z_0}{L}\right)\right] \tag{2.42}$$

gdzie wielkości ψ_M wprowadzają do profilu wiatru poprawkę w formie:

$$\psi_M = \begin{cases} ln\left[\left(\frac{1+x^2}{2}\right)\left(\frac{1+x}{2}\right)^2\right] - 2tan^{-1}x + \frac{\pi}{2}, & dla \quad \frac{z}{L} < 0 \qquad (2.43a) \\ -5\frac{z}{L}, & dla \quad \frac{z}{L} \geq 0 \qquad (2.43b) \end{cases}$$

gdzie x wyraża się jako:

$$x = \left(1 - 15\left(\frac{z}{L}\right)\right)^{1/4} \tag{2.44}$$

Analogicznie, poprzez całkowanie pozostałych równań 2.31 i 2.32 z odpowiednio ϕ_h i ϕ_w otrzymuje się:

$$\theta(z) - \theta(z_{0H}) = \frac{\theta_*}{\kappa} \cdot \left[ln\frac{z}{z_{0H}} - \psi_H\left(\frac{z}{L}\right) + \psi_H\left(\frac{z_{0H}}{L}\right)\right] \tag{2.45}$$

$$q(z) - q(z_{0W}) = \frac{q_*}{\kappa} \cdot \left[ln\frac{z}{z_{0W}} - \psi_W\left(\frac{z}{L}\right) + \psi_W\left(\frac{z_{0W}}{L}\right)\right] \tag{2.46}$$

gdzie ψ_H i ψ_W przedstawiają się w formie [23]:

$$\psi_H = \psi_W = \begin{cases} 2ln(\frac{1+x^2}{2}), & dla \quad \frac{z}{L} < 0 \quad (2.47a) \\ -5\frac{z}{L}, & dla \quad \frac{z}{L} \geq 0 \quad (2.47b) \end{cases}$$

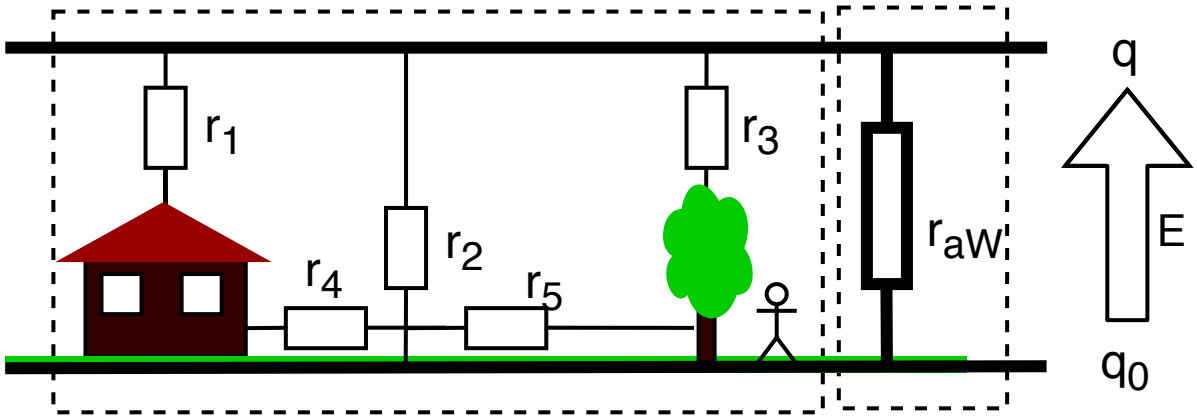
Wielkości z_{0H} i z_{0W} to odpowiednio współczynnik szorstkości dla temperatury i współczynnik szorstkości dla wilgotności. Poprzez analogię do z_0, współczynniki te są interpretowane jako wysokości nad powierzchnią, na których temperatura i wilgotność są równe temperaturze i wilgotności tej powierzchni [18].

W celach ułatwienia analizy przepływu strumieni turbulencyjnych wprowadza się, w analogi do prawa Ohma, współczynniki oporu aerodynamicznego r_{aM}, r_{aH} i r_{aW} (rysunek 4). Strumienie turbulencyjne przy powierzchni mogą wtedy zostać określone w sposób:

$$\tau_0 = \rho \cdot \frac{(u_0 - u)}{r_{aM}} \quad (2.48)$$

$$H_0 = -c_p \cdot \rho \cdot \frac{(\theta_0 - \theta)}{r_{aH}} \quad (2.49)$$

$$E_0 = -l \cdot \rho \cdot \frac{(q_0 - q)}{r_{aW}} \quad (2.50)$$



Rysunek 4: Obrazowe przedstawienie pojęcia oporu aerodynamicznego na przykładzie strumienia ciepła utajonego (opracowanie własne na podstawie [18]).

Czynniki $1/r_{aM}$, $1/r_{aH}$ i $1/r_{aW}$ są odpowiednikami przewodności elektrycznej. Parametryzuje się je jako iloczyn prędkości wiatru i współczynników *bulk transfer*:

$$r_{aM}^{-1} = u \cdot C_D \quad (2.51)$$

$$r_{aH}^{-1} = u \cdot C_H \quad (2.52)$$

$$r_{aW}^{-1} = u \cdot C_E \quad (2.53)$$

Na podstawie teorii podobieństwa Monina–Obuchowa współczynniki bulk transfer przybierają postać [18]:

$$C_D = \kappa^2 \left[ln\frac{z}{z_0} - \psi_M\left(\frac{z}{L}\right) + \psi_M\left(\frac{z_0}{L}\right)\right]^{-2} \tag{2.54}$$

$$C_H = \kappa^2 \left[ln\frac{z}{z_0} - \psi_M\left(\frac{z}{L}\right) + \psi_M\left(\frac{z_0}{L}\right)\right]^{-1} \cdot \left[ln\frac{z}{z_{0H}} - \psi_H\left(\frac{z}{L}\right) + \psi_H\left(\frac{z_{0H}}{L}\right)\right]^{-1} \tag{2.55}$$

$$C_E = \kappa^2 \left[ln\frac{z}{z_0} - \psi_M\left(\frac{z}{L}\right) + \psi_M\left(\frac{z_0}{L}\right)\right]^{-1} \cdot \left[ln\frac{z}{z_{0W}} - \psi_W\left(\frac{z}{L}\right) + \psi_W\left(\frac{z_{0W}}{L}\right)\right]^{-1} \tag{2.56}$$

Ostatecznie, powierzchniowe strumienie turbulencyjne pędu, ciepła jawnego i ciepła utajonego można przedstawić jako:

$$\tau_0 = \rho \cdot u^2 \cdot C_D \tag{2.57}$$

$$H_0 = -c_p \cdot \rho \cdot u \cdot C_H \cdot (\theta_0 - \theta) \tag{2.58}$$

$$E_0 = -l \cdot \rho \cdot u \cdot C_E \cdot (q_0 - q) \tag{2.59}$$

3 Metodyka pomiarowa turbulencyjnych strumieni ciepła

3.1 Metoda gradientowa

Najprostszą metodą pomiarową strumieni turbulencyjnych jest metoda oparta na pomiarach prędkości wiatru, temperatury powietrza i wilgotności powietrza na dwóch różnych poziomach z_1 i z_2 mierzonych od powierzchni ziemi. Zastąpienie temperatury potencjalnej temperaturą absolutną (wyrażoną w K), tak jak wspominano w poprzednim rozdziale, w przyziemnej warstwie atmosfery jest procedurą dopuszczalną. Konieczna jest również znajomość parametrów terenu w postaci przesunięcia płaszczyzny zerowej pionowego profilu wiatru d. Współczynnik ten należy wyznaczyć eksperymentalnie na podstawie pomiarów prędkości wiatru na różnych wysokościach w warunkach neutralnej stratyfikacji atmosfery poprzez dopasowanie do tych wyników krzywej logarytmicznej o równaniu 2.41 (jednocześnie, wyznaczy się parametr szorstkości z_0).

W kolejnym kroku, na podstawie wzoru 3.60, wyznacza się gradientową liczbę Richardsona, uwzględniając w niej współczynnik przesunięcia d:

$$Ri(z_s - s) = \frac{g}{T_0} \cdot (z_s - d) \cdot ln\frac{z_2 - d}{z_1 - d} \cdot \frac{\Delta T}{(\Delta u)^2} \tag{3.60}$$

A na jej podstawie, oraz zależności 2.35, 2.33 i 2.34 wyznacza się parametr ζ (przy założeniu

$\gamma_1 = \gamma_2 = 15$, $\beta_1 = \beta_2 = 5$ i $P_t = 1$) [28]:

$$\zeta = \begin{cases} Ri(z_s - d), & dla \quad Ri(z_s - d) < 0 \\ \frac{Ri(z_s - d)}{1 - 5Ri(z_s - d)}, & dla \quad Ri(z_s - d) \geq 0 \end{cases} \tag{3.61}$$

Następnie, korzystając z wyrażeń 2.33 i 2.34 oraz powyższych wartości współczynników w nich występujących, wyznacza się funkcje podobieństwa ϕ_M, ϕ_H i ϕ_W. Wykorzystując zależności 2.30 - 2.32 i 2.36 - 2.38 oraz po zastąpieniu pionowych gradientów skończonymi przyrostami odpowiednich wielkości ($\frac{\partial A}{\partial z} \mapsto \frac{\Delta A}{\Delta z}$), przedstawić można formuły na wartości strumieni turbulencyjnych [28]:

$$\tau = \rho \cdot \left[\frac{\kappa \cdot \Delta u}{\phi_M(\zeta) \cdot ln \frac{z_2 - d}{z_1 - d}} \right]^2 \tag{3.62}$$

$$H = -c_p \cdot \rho \cdot \frac{\kappa^2 \cdot \Delta u \cdot \Delta T}{\phi_M(\zeta) \cdot \phi_H(\zeta) \left(ln \frac{z_2 - d}{z_1 - d} \right)^2} \tag{3.63}$$

$$E = -l \cdot \rho \cdot \frac{\kappa^2 \cdot \Delta u \cdot \Delta q}{\phi_M(\zeta) \cdot \phi_W(\zeta) \left(ln \frac{z_2 - d}{z_1 - d} \right)^2} \tag{3.64}$$

Przedstawiona metoda może być stosowana wyłącznie dla parametru ζ z przedziału $-5 < \zeta < 1$, gdyż dla takich wartości tego parametru wyznaczone są funkcje podobieństwa.

3.2 Metoda bulk transfer

Metoda ta w istocie również jest metodą opartą na pomiarach gradientowych, jednak opisaną z wykorzystaniem zależności opartych o współczynniki bulk transfer. W pierwotnym założeniu do obliczeń, oprócz pomiarów temperatury, wilgotności i prędkości wiatru na pewnej wysokości z, wymagana jest znajomość wartości temperatur i wilgotności powietrza na wysokości współczynników z_{0H} i z_{0W}. Pomiary te są jednak utrudnione, ponieważ nie można w sposób precyzyjny określić wartości współczynników szorstkości temperatury i wilgotności, oraz nie istnieje jednoznaczna definicja powierzchni względem której te pomiary miałyby zostać wykonane. W przypadku obszarów wysoce zurbanizowanych hipotetyczną powierzchnią mogą być zarówno powierzchnia ziemi, jaki i powierzchnie elementów zabudowy. Z tego powodu temperaturę tę często zastępuje się pomiarem temperatury radiacyjnej gruntu, a współczynnik szorstkości określa na podstawie różnych oszacowań. Problematyczne są również pomiary wilgotności na wysokości z_{0W}, które również muszą zostać oszacowane [18]. Jedną z możliwych metod ominięcia ich wyznaczania jest wykonywanie pomiarów na pewnej wysokości nad poziomem gruntu i obliczenie na podstawie wzorów 2.45 i 2.46 zależności opisujących zmiany temperatur i wilgotności

wraz z wysokością. Na tej podstawie i z wykorzystaniem zależności 2.27 - 2.29, oraz przy założeniu znajomości parametrów d i z_0 profilu wiatru, zapisać można formuły na charakterystyki prędkości wiatru, temperatury i wilgotności:

$$u_* = \frac{\kappa \cdot u}{ln\frac{z_u - d}{z_0} - \psi_M\left(\frac{z_u - d}{L}\right) + \psi_M\left(\frac{z_0}{L}\right)} \tag{3.65}$$

$$\theta_* = \frac{\kappa \cdot (T_2 - T_1)}{ln\frac{z_2 - d}{z_1 - d} - \psi_H\left(\frac{z_2 - d}{L}\right) + \psi_H\left(\frac{z_1 - d}{L}\right)} \tag{3.66}$$

$$q_* = \frac{\kappa \cdot (q_2 - q_1)}{ln\frac{z_2 - d}{z_1 - d} - \psi_W\left(\frac{z_2 - d}{L}\right) + \psi_W\left(\frac{z_1 - d}{L}\right)} \tag{3.67}$$

gdzie: u - prędkość wiatru na wysokości z_u; T_1, T_2 - temperatury absolutne na wysokościach z_1 i z_2 ($z_2 > z_1$); q_1, q_2 - wilgotności właściwe na wysokościach z_1 i z_2. Długość Obuchowa można przedstawić wtedy jako:

$$L = \frac{u_*^2 \cdot \frac{T_1 + T_2}{2}}{\kappa \cdot g \cdot \theta_*} \tag{3.68}$$

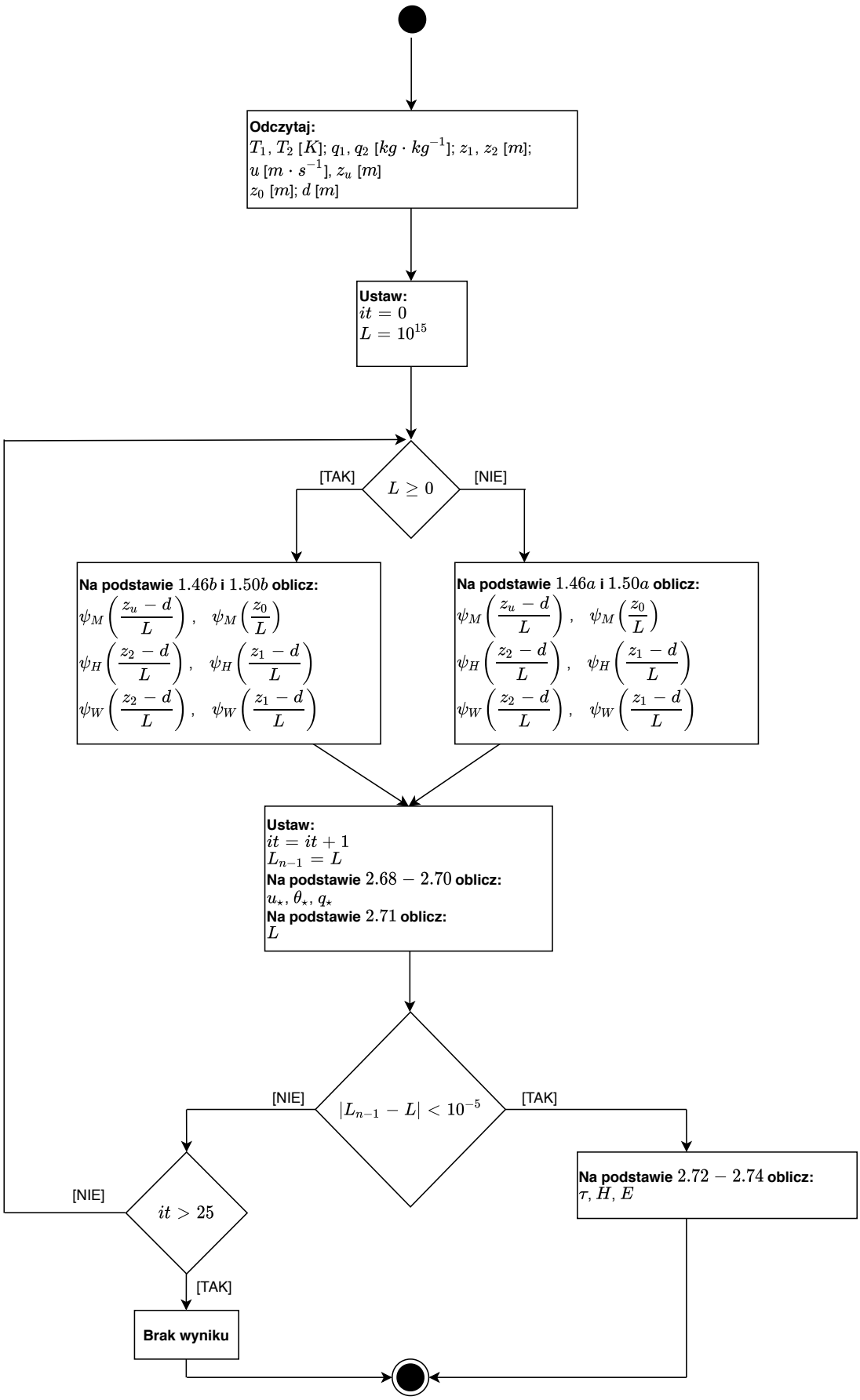
a turbulencyjne strumienie pędu, ciepła jawnego i utajonego poprzez:

$$\tau = \rho \cdot u_*^2 \tag{3.69}$$

$$H = -\rho \cdot c_p \cdot u_* \cdot \theta_* \tag{3.70}$$

$$E = -\rho \cdot l \cdot u_* \cdot q_* \tag{3.71}$$

Podstawowym problemem przedstawionej metody oszacowywania jest to, że opiera się ona na układzie czterech równań 3.65 - 3.68, w których równania 3.65 - 3.67 zależą od równania 3.68, które to równanie zależy od równań 3.65 - 3.67. Nie jest więc możliwe przedstawienie analitycznego ich rozwiązania. Możliwe jest jednak podjęcie próby rozwiązania tego układu metodą iteracyjną ([23], [25], [26]). Polega ona na założeniu w pierwszym kroku warunków obojętnej stratyfikacji atmosfery, dla których parametr $\zeta = 0$. W takim wypadku długość Obuchowa L dąży do nieskończoności, a numerycznie można ją przykładowo przedstawić jako wartość $L = 10^{15}\ m$. Następnie należy obliczyć charakterystyki 3.65 - 3.67, a na ich podstawie nową wartość parametru L. Procedurę tę należy przeprowadzać do momentu ustabilizowania się wartości L (przykładowo: $|L_{n-1} - L| < 10^{-5}\ m$). Końcowo, za pomocą nowo otrzymanych wartości charakterystyk, oblicza się strumienie na podstawie wzorów 3.69 - 3.71. Graficzne przedstawienie niniejszego algorytmu obliczeniowego przedstawia rysunek 5.

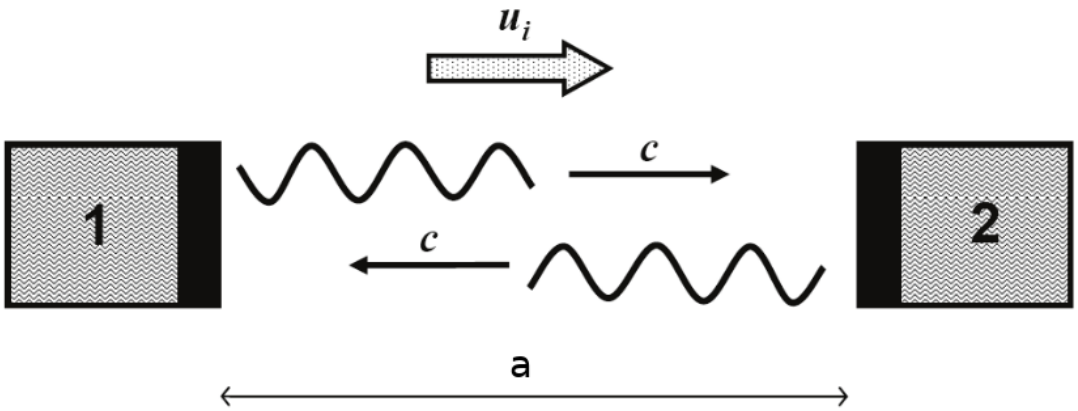


Rysunek 5: Schemat algorytmu metody obliczeniowej bulk transfer

3.3 Metoda kowariancji wirów

Obecnie najpopularniejszą metodą pomiarów strumieni turbulencyjnych ciepła jest metoda kowariancji wirów (z ang. *eddy covariance method*). Opiera się ona na pomiarze strumieni turbulencyjnych wprost z definicji na podstawie wzorów 2.15 - 2.16.

W celu pomiarów strumieni ciepła jawnego wykorzystuje się anemometry akustyczne, których działanie polega na pomiarze prędkości wiatru i temperatury za pomocą sygnałów akustycznych, których prędkość rozchodzenia się w powietrzu zależy właśnie od tych wielkości. W praktyce równanie 2.15 wymaga aby pomiary te były wykonywane z wysoką częstotliwością, co przekłada się na częstotliwości przynajmniej $10Hz$ [18].



Rysunek 6: Zasada działania anemometru akustycznego (źródło obrazu: [18]).

W anemometrach akustycznych impulsy dźwiękowe emitowane są między dwoma przetwornikami (rysunek 6). Czasy przebycia drogi między nimi można przedstawić jako:

$$t_1 = \frac{a}{c + u_i} \tag{3.72}$$

$$t_2 = \frac{a}{c - u_i} \tag{3.73}$$

gdzie: t_1 - czas przebycia drogi a między przetwornikiem 1 a 2, t_2 - czas przebycia drogi a między przetwornikiem 2 a 1, u_i - składowa prędkości wiatru między przetwornikami. Na ich podstawie wyznaczyć można odpowiednią składową prędkości wiatru. Wiedząc, że prędkość dźwięku w powietrzu określa się jako:

$$c = \frac{a}{2} \cdot \left(\frac{1}{t_1} + \frac{1}{t_2} \right) \tag{3.74}$$

wyznaczyć można temperaturę powietrza na podstawie zależności [18]:

$$c^2 = \gamma_d \cdot R_d \cdot T_s = \gamma_d \cdot R_d \cdot T \cdot (1 + 0.51q) \tag{3.75}$$

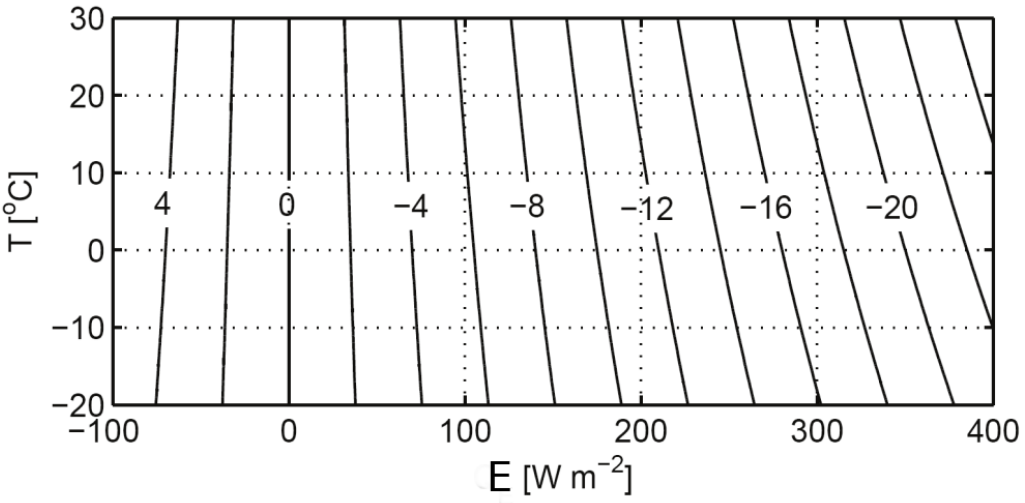
gdzie: R_d - stała gazowa powietrza suchego, T_s - temperatura zależna od temperatury rzeczywistej poprzez zależność: $T_s = T \cdot (1 + 0.51q)$, $\gamma_d = \frac{c_p \cdot (1+0.93 \cdot q)}{c_v \cdot (1+0.84q)}$, c_v - ciepło właściwe powietrza w stałej objętości.

Rysunek 7: Przykład anemometru akustycznego wykonującego pomiary w trzech płaszczyznach.

Ostatecznie, po przyjęciu przybliżenia $T \approx (1 - 0.51 \cdot q) \cdot T_s$, oraz $\frac{T}{T_s} \approx 1$, strumień ciepła jawnego przedstawić można za pomocą wyrażenia:

$$H = H_s - 0.51 \cdot \bar{T} \cdot \frac{c_p}{l} \cdot E \tag{3.76}$$

gdzie: H_s - strumień ciepła wyznaczony na podstawie temperatury T_s i definicji 2.15. Rzeczywisty strumień ciepła jawnego mierzony przez anemometr, jest więc powiązany ze strumieniem ciepła utajonego. Popularną techniką przeliczania tych zależności na strumienie rzeczywiste jest wykorzystanie przybliżonych wartości czynników $H - H_s$. Przykładowe ich wartości przedstawia nomogram na rysunku 8.



Rysunek 8: Przykładowy nomogram przedstawiający wartości czynników $H - H_s$ w zależności od temperatury i strumienia ciepła utajonego (źródło obrazu: [18]).

W celu pomiarów strumieni ciepła utajonego, związanego z szybkozmiennymi pomiarami wilgotności powietrza, wykorzystuje się analizatory gazowe. Opierają się one na prawie Bouguera–Lamberta–Beera, które mówi, że wyemitowane promieniowanie o natężeniu I_0 przechodząc przez ośrodek o gęstości ρ_o, na drodze x, jest osłabiane do natężenia I zgodnie z równaniem:

$$I = I_0 \cdot exp(-k_\lambda \cdot \rho_o \cdot x) \tag{3.77}$$

gdzie: k_λ - współczynnik absorpcji fali o długości λ. Wykorzystując długości fal, które tłumione są wyłącznie przez parę wodną (na przykład $= 121.56nm$) wyznaczyć można bezwzględną wilgotność powietrza ρ_{H_2O}:

$$\rho_{H_2O} = (k_\lambda \cdot x)^{-1} \cdot ln(I_0/I) \tag{3.78}$$

a następnie strumień ciepła utajonego, na postawie zależności:

$$E = l \cdot (\overline{w \cdot \rho_{H_2O}}) \tag{3.79}$$

W praktyce, z powodu konieczności wykonywania wielu uśrednień, przybliżeń i uwzględniania wielu innych poprawek, w celu wyliczania strumieni turbulencyjnych wykorzystuje się zautomatyzowane oprogramowanie (na przykład EddyPro [8]).

4 Budowa systemu pomiarowego

4.1 Określenie celu

Celem systemu ma być zbieranie danych meteorologicznych w postaci temperatury i wilgotności powietrza na kilku wysokościach w celu wyznaczania metodą gradientową strumieni turbulencyjnych ciepła jawnego i utajonego w planetarnej warstwie granicznej atmosfery ziemskiej. Dane dotyczące prędkości wiatru uzyskiwane będą ze stacji meteorologicznej znajdującej się na dachu budynku Wydziału Fizyki i Informatyki Stosowanej AGH [39].

4.2 Specyfikacja wymagań systemowych

Przed rozpoczęciem budowy systemu przystąpiono do zebrania i usystematyzowania jego oczekiwanych wymagań funkcjonalnych i pozafunkcjonalnych. Podjęto również próbę określenia możliwych trudności, które mogłyby wystąpić w trakcie procesu jego tworzenia. Całościową analizę przedstawia tabela 1 i 2.

Tabela 1: Analiza wymagań funkcjonalnych

ID	Nazwa	Wymaganie	trudności
1	Pomiar parametrów meteorologicznych	System powinien zapewniać możliwość pomiaru temperatury powietrza, wilgotności powietrza i ciśnienia atmosferycznego.	brak
2	Czas pracy	System powinien mieć możliwość pracy ciągłej.	brak
3	Zapis danych	System powinien mieć możliwość zapisywania odczytywanych parametrów meteorologicznych wraz z datą i godziną odczytu.	brak
4	Format zapisu danych	System powinien zapisywać dane w postaci plików CSV na karcie microSD. Dane z każdego dnia powinny być zapisywane w osobnym pliku, którego nazwa powinna zawierać rok, miesiąc i dzień.	brak
5	Częstotliwość pomiarów	System powinien odczytywać i zapisywać aktualne parametry meteorologiczne przynajmniej raz na minutę.	brak
6	Sygnalizacja poprawności działania systemu	System powinien posiadać właściwość sygnalizacji poprawności jego działania. W przypadku działania nieprawidłowego, powinien sygnalizować który jego element działa nieprawidłowo.	Potencjalny problem detekcji nieprawidłowości działania.
7	Watchdog	System powinien posiadać możliwość samodetekcji stanu zawieszenia. W wypadku jego wystąpienia powinien doprowadzić do restartu.	brak

Tabela 2: Analiza wymagań pozafunkcjonalnych

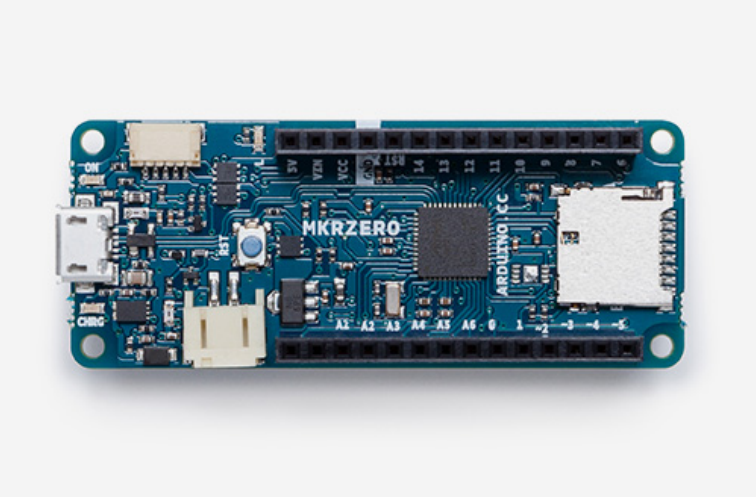
ID	Nazwa	Wymaganie	trudności
1	Poziomy pomiarów	System powinien odczytywać parametry meteorologiczne na przynajmniej trzech różnych wysokościach na maszcie pomiarowym. Różnica skrajnych wysokości pomiarów powinna wynosić około 20 metrów.	Zapewnienie odpowiedniej jakości transmisji danych i eliminacja zakłóceń.
2	Bezpieczeństwo	System powinien posiadać zabezpieczenia przeciwzwarciowe i zabezpieczenia przed odwrotną polaryzacją zasilania.	brak
3	Miejsce pomiarów	System powinien być zamontowany na maszcie znajdującym się na dachu budynku Wydziału Fizyki i Informatyki Stosowanej Akademii Górniczo-Hutniczej. Sterownik systemu może być umieszczony w pomieszczeniu znajdującym się pod masztem.	brak
4	Odporność na warunki atmosferyczne	System powinien być odporny na warunki meteorologiczne panujące w miejscu pomiaru niezależnie od pory roku.	Zapewnienie odpowiedniej szczelności złącz elektrycznych i ochrona elektroniki.

4.3 Podzespoły

Na podstawie szczegółowej analizy wcześniej wspomnianych wymagań, do budowy systemu wybrano podzespoły opisane w niniejszej części pracy.

4.3.1 Arduino MKRZero

Jako bazę systemu postanowiono wybrać moduł Arduino MKRZero (Rysunek 9). Posiada on wbudowany czytnik kart microSD oraz jest kompaktowych rozmiarów, co czyni go idealnym do zastosowania w projekcie niniejszego systemu.



Rysunek 9: Płytka Arduino MKRzero [29]

Zgodnie z dokumentacją producenta moduł posiada następujące parametry techniczne [29]:

- 32-bitowy mikroprocesor SAMD21 Cortex-M0+ taktowany zegarem 48 MHz
- Zasilanie napięciem 5V
- Magistrale I^2C, $UART$, SPI, USB
- Napięcie poziomów logicznych: 3.3V

Ponadto, jak w przypadku każdego modułu z rodziny Arduino, producent dostarcza platformę programistyczną, wliczając w to własny język programowania z bogatą i szczegółową dokumentacją oraz środowisko programistyczne ułatwiające programowanie modułów [6]. Sprawia to, że powyższe parametry techniczne, z punktu widzenia wymagań systemowych, są całkowicie wystarczające. Jedynym problemem, który występuje w przypadku wykorzystywania tego modułu, to napięcie poziomów logicznych, na których on pracuje. Zdecydowana większość dostępnych na rynku sensorów i podzespołów operuje na napięciu poziomów logicznych wynoszącym 5V. Z tego powodu konieczne może być wprowadzenie konwerterów poziomów logicznych do budowanych układów.

4.3.2 Sensor BME280

Układ BME280 to sensor wilgotności, ciśnienia i temperatury produkowany przez firmę BOSCH [9]. Jego niewielkie rozmiary ($2.5mm \times 2.5mm$) oraz szeroka funkcjonalność powodują, że jest na szeroką skalę używany w przemyśle oraz przez producentów sprzętu elektronicznego. Na rynku występuje też wiele gotowych modułów przystosowanych do pracy bezpośrednio z

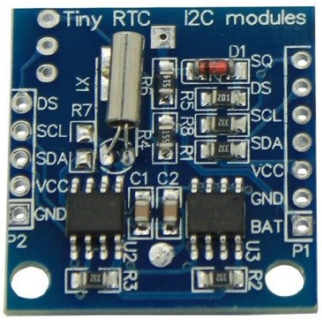
platformą Arduino (Rysunek 10). Ich producenci dostarczają też gotowe biblioteki ułatwiające komunikację z sensorem [1].

Rysunek 10: Wykorzystany w niniejszej pracy układ BME280 przystosowany do pracy z platformą Arduino. Producent: Waveshare. (Źródło: strona internetowa producenta [35])

BME280 może komunikować się poprzez interfejsy I^2C oraz SPI. Typowe napięcie pracy zalecane przez producenta 3.3V, jednak niektórzy producenci gotowych modułów przystosowują je dodatkowo do pracy z napięciem 5V. Taki też moduł wykorzystany został w niniejszej pracy.

4.3.3 Zegar czasu rzeczywistego DS1307

Do odczytywania aktualnego czasu wykorzystano zegar czasu rzeczywistego (RTC) oparty na układzie DS1307 [11]. Posiada on niezależne podtrzymujące zasilanie bateryjne o napięciu $3V$, więc nawet po odłączeniu zewnętrznego zasilania podtrzymywana jest jego praca. DS1307 może być zasilany napięciem z zakresu $4.5V$ - $5.5V$. Układ dostarcza możliwość komunikacji z nim poprzez magistralę I^2C. Tak jak w przypadku sensora BME280, na rynku występuje wiele modułów przystosowanych już do pracy bezpośrednio z Arduino.



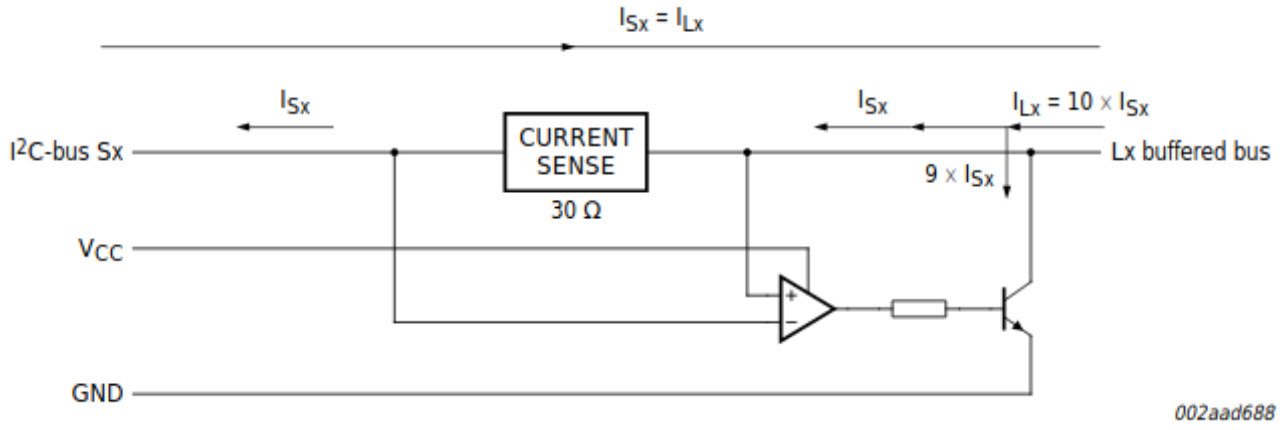
Rysunek 11: Wykorzystany w niniejszej pracy układ DS1307 przystosowany do pracy z platformą Arduino. Producent: Nieznany. (Źródło obrazu: https://abc-rc.pl/product-pol-6190-Modul-czasu-RTC-DS1307-zegar-czasu-rzeczywistego-Arduino.html)

4.3.4 Ekstender magistrali I^2C P82B715

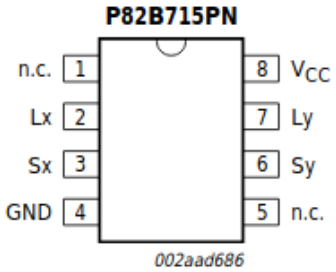
Magistrala I^2C jest szeregową i dwukierunkową magistralą służącą do przesyłu danych cyfrowych. Występują w niej dwie linie oznaczane jako SDA - linia danych, oraz SCL - linia zegarowa. Jako, że zarówno moduł Arduino MKRZero, sensory BME280, oraz zegar czasu rzeczywistego DS1307 wspierają komunikację poprzez tę magistralę, postanowiono to właśnie ją wybrać jako sposób komunikacji w budowanym układzie. Ponadto, interfejs I^2C pozwala na podłączenie do jednej linii kilku urządzeń. Każde urządzenie powinno posiadać jednak swój osobny adres.

Pierwotnie I^2C zaprojektowana została do transmisji danych między układami znajdującymi się w niewielkiej odległości od siebie, a najlepiej wbudowanych w tę samą płytkę drukowaną PCB. Całkowita odległość transmisji ograniczona jest przez maksymalną pojemność linii wynoszącą $400pF$ [17] [24]. Przy pojemnościach 100 pF na metr długości przewodu, mogących pojawiać się na linii w trakcie komunikacji poprzez I^2C [14], maksymalna odległość stabilnej transmisji wynosi więc około 4 metry. W budowanym w niniejszej pracy układzie konieczna jest transmisja na znacznie większe odległości. W tym celu wykorzystany więc został układ P82B715 [14], którego zadaniem jest zwiększenie maksymalnej dopuszczalnej pojemności linii zegarowej i sygnałowej magistrali do 4 nF. Realizowane jest to poprzez 10 krotne wzmocnienie prądowe sygnałów. Konieczne jest zastosowanie układu przy każdym urządzeniu wpiętym w magistralę komunikacyjną, a jego działanie jest dwukierunkowe. Schemat zastępczy ukła-

du wzmacniającego jedną linię przedstawia rysunek 12. Wyprowadzenia układu przedstawia rysunek 13 a ich opis tabela 3.



Rysunek 12: Schemat zastępczy jednej linii układu P82B715 (Źródło obrazu: Dokumentacja techniczna układu P82B715 [14]).

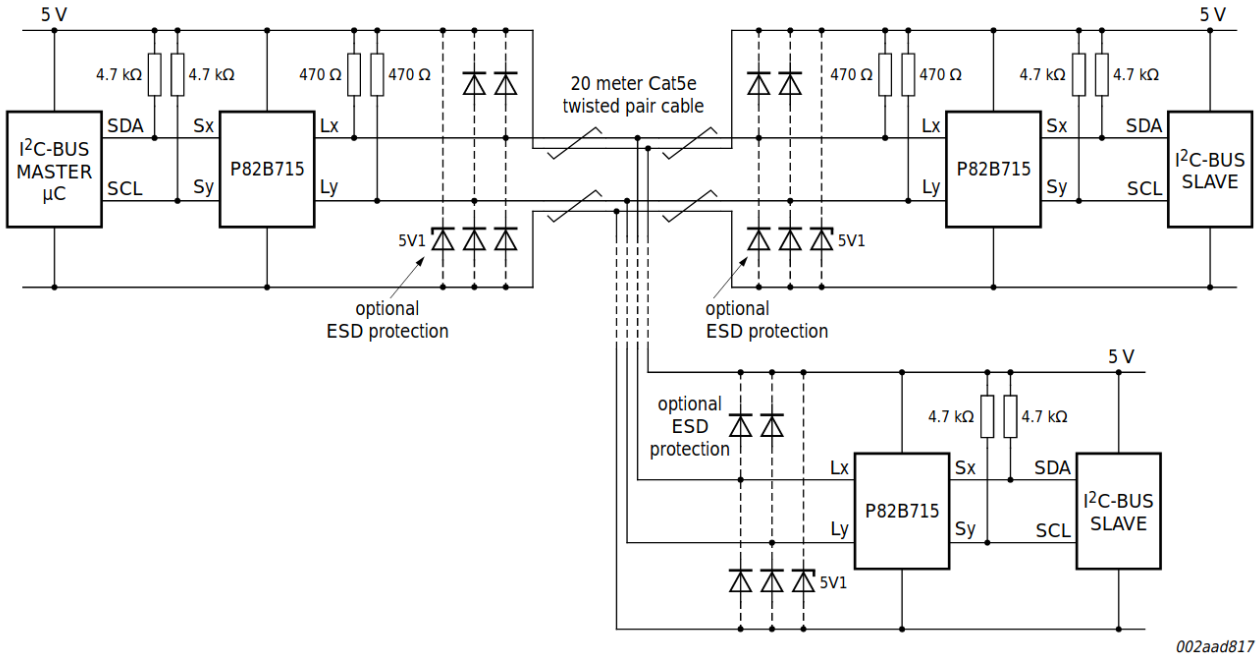


Rysunek 13: Wyprowadzenia układu P82B71. (Źródło obrazu: Dokumentacja techniczna układu P82B715 [14]).

Tabela 3: Opis wyprowadzeń układu P82B715 [14]

Pin	Symbol	Opis
1	n.c.	nie podłączony
2	Lx	Buforowana linia magistrali, LDA lub LCL
3	Sx	Linia magistrali I^2C, SDA lub SCL
4	GND	masa zasilania
5	n.c.	nie podłączony
6	Sy	Linia magistrali I^2C, SDA lub SCL
7	Ly	Buforowana linia magistrali, LDA lub LCL
8	Vcc	zasilanie

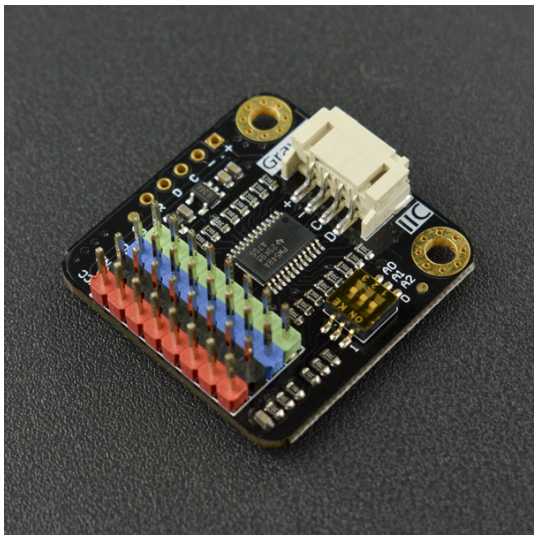
Dokumentacja techniczna układu dostarcza również schemat typowego zastosowania z proponowanymi wartościami rezystorów pull-up o wartości 470Ω po stronie magistrali buforowanej i $4.7k\Omega$ po stronie magistrali I^2C.



Rysunek 14: Schemat typowego zastosowania układu P82B715. (Źródło obrazu: Dokumentacja techniczna układu P82B715 [14]).

4.3.5 Multiplekser I^2C TCA9548A

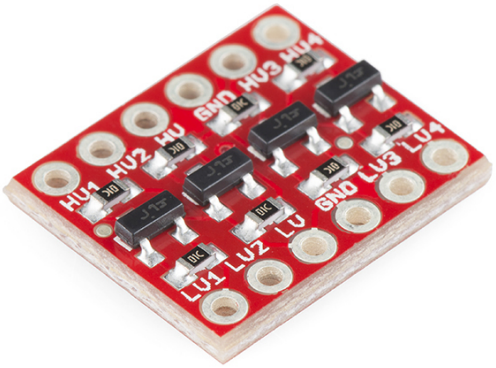
Pomimo, że magistrala I^2C dopuszcza podłączenie wielu urządzeń do jednej linii, to urządzenia te muszą posiadać osobne adresy, które są określane przez ich własne kontrolery i nie zawsze mogą być zmienione. W przypadku wystąpienia na linii konfliktu dwóch takich samych adresów, system nie może działać prawidłowo. W celu przyłączenia do linii urządzeń o takim samym adresie można jednak wykorzystać multipleksowanie linii przyłączeniowych. W wykorzystywanych w niniejszym projekcie układach Waveshare [35], producent dopuszcza wybór adresu tylko z puli dwóch, więc w przypadku zastosowania trzech sensorów wystąpiłby konflikt dwóch z nich. Z tego powodu postanowiono, że każdy z układów przyłączany będzie do magistrali I^2C poprzez multiplekser. Zastosowany w projekcie gotowy moduł multipleksera DFRobot Gravity DFR0576 [31] (rysunek 16) oparty jest na układzie TCA9548A [16] i przystosowany jest bezpośrednio do podłączenia do Arduino. Producent dostarcza też bibliotekę ułatwiającą jego wykorzystanie w projektach [2].



Rysunek 15: Moduł multipleksera DFRobot Gravity DFR0576 (Źródło obrazu: strona internetowa producenta [33]).

4.3.6 Konwerter poziomów logicznych SparkFun BOB-12009

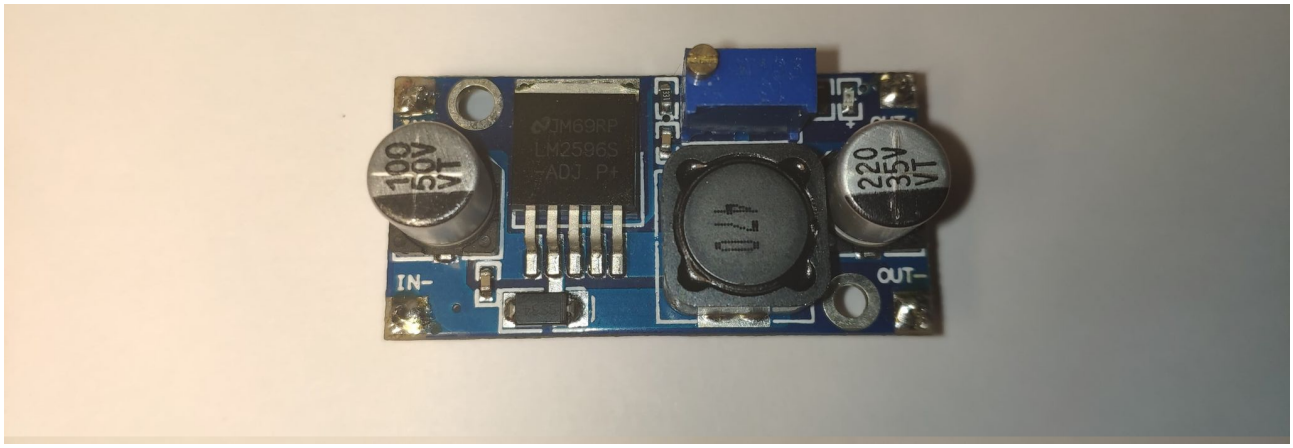
Gdy zachodzi potrzeba komunikacji cyfrowej między podzespołami posiadającymi różne napięcia poziomów logicznych transmisji, pojawia się konieczność dwukierunkowej konwersji tych poziomów. W tym celu stosuje się konwertery poziomów logicznych. Moduł SparkFun BOB-12009 składa się z czterech równolegle ułożonych układów opartych o tranzystory BSS138 [10].



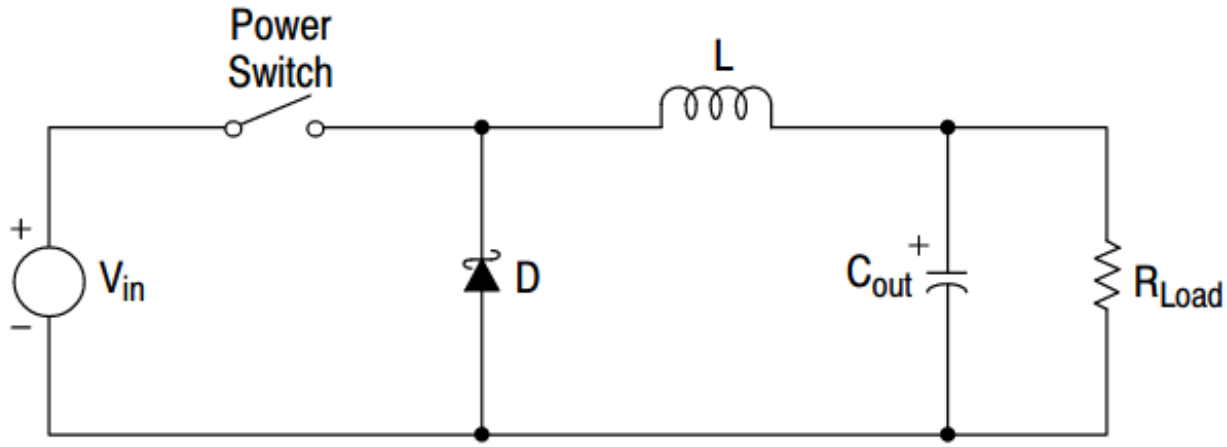
Rysunek 16: Moduł konwertera poziomów logicznych (Źródło obrazu: strona internetowa producenta [33]).

4.3.7 Przetwornica step-down

W celu obniżenia napięcia zasilania do napięcia $5V$ wymaganego do zasilenia podzespołów bazowych wykorzystano moduł oparty o układ LM2596 [12]. Rysunek 17 przedstawia jeden z przykładowych gotowych modułów szeroko dostępnych w sprzedaży, popularnie nazywanych przetwornicami step-down.



Rysunek 17: Moduł obniżający napięcie oparty na układzie LM2596.



Rysunek 18: Schemat prostego konwertera obniżającego napięcie (Źródło obrazu: Dokumentacja techniczna układu LM2596 [12]).

Moduły te realizują klasyczną metodę impulsowego obniżania napięcia, której schemat przedstawia rysunek 18. Działanie takiego konwertera podzielić można na dwa etapy. W pierwszym etapie przełącznik zasilania jest zwarty (przewodzi), więc prąd przez cewkę L przepływa w sposób liniowo narastający (do pewnej wartości szczytowej), a dioda D znajduje się w stanie zaporowym. W drugim etapie następuje rozwarcie przełącznika zasilania. Opadający prąd na

cewce powoduje wyindukowanie w niej siły elektromotorycznej o przeciwnym znaku, więc dioda zaczyna przewodzić w kierunku do obciążenia R_{Load}. Kondensator C_{Out} pełni tutaj rolę filtra dolnoprzepustowego, zmniejszającego tętnienia napięcia wyjściowego. Dobierając odpowiednio częstotliwość zwierania i rozwierania przełącznika uzyskać można dowolną wartość napięcia wyjściowego (większą od 0V i mniejszą od napięcia zasilania). W praktyce jako przełącznik wykorzystuje się układ tranzystora pracującego jako klucz, sterowany przez generator sygnału PWM o odpowiednim współczynniku wypełniania.

4.3.8 Przewody sygnałowe i zasilające

BiTsensor PE-PVC Blue 2x2x22AWG

Jako linię przesyłu danych wykorzystano ekranowany i parowany kabel transmisyjny producenta Bitner [20] (rysunek 19). Posiada on cztery miedziane żyły o przekroju $0.34mm^2$. Linia czarna i czerwona dedykowana jest do przesyłu zasilania, natomiast biała i zielona do przesyłu transmisji cyfrowych. Ekranowanie par chroni przesyłane sygnały przed wpływem zewnętrznych zakłóceń elektromagnetycznych. Kabel posiada powłokę odporną na promieniowanie UV i jest dopuszczony do użytku zewnętrznego.

Rysunek 19: Kabel BiTsensor PE-PVC Blue 2x2x22AWG (Źródło obrazu: karta katalogowa produktu [20]).

Helukabel SiHF 2x0.5

Jako linię zasilającą wykorzystano bezhalogenowy wielożyłowy przewód producenta Helukabel [21] (rysunek 20). Posiada on dwie miedziane żyły o przekroju $0.5mm^2$, ognioodporną, silikonową powłokę oraz jest dopuszczony do użytku zewnętrznego.

Rysunek 20: Kabel Helukabel SiHF 2x0.5

4.3.9 Osłony radiacyjne sensorów, puszki instalacyjne i złącza przemysłowe

Osłona radiacyjna TFA 98.1114.02

W celu zapewnienia ochrony modułów sensorów przed bezpośrednim działaniem czynników atmosferycznych oraz promieniowania słonecznego, zostały one umieszczone w osłonie radiacyjnej TFA 98.1114.02 [22](rysunek 21).

Rysunek 21: Osłona radiacyjna TFA 98.1114.02 (Źródło obrazu: karta katalogowa produktu [22]).

Puszka instalacyjna Spelsberg Abox 025-L

Moduły elektroniczne, zapewniające komunikację między sensorami a sterownikiem systemu, umieszczone zostały w puszkach instalacyjnych producenta Spelsberg [34] (rysunek 22). Spełniają one normę szczelności IP65. Oznacza to całkowitą pyłoszczelność i wysoką ochronę przed wodą i czynnikami atmosferycznymi.

Rysunek 22: Puszka instalacyjna Spelsberg Abox 025-L (Źródło obrazu: strona internetowa producenta [34]).

Złącza przemysłowe WEiPU SP13

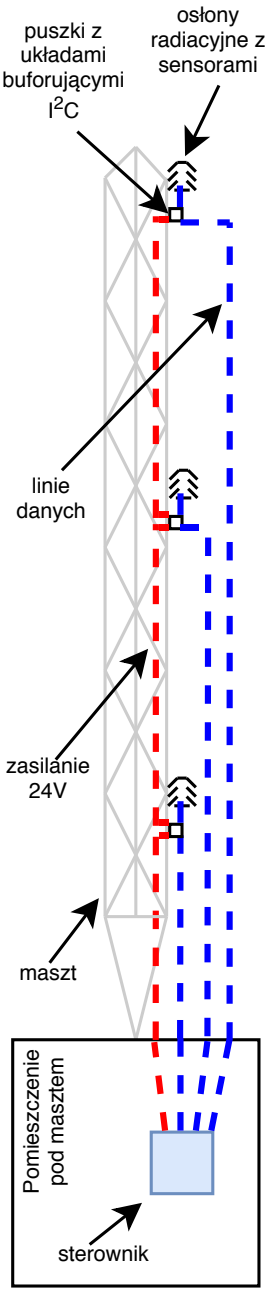
Połączenia kabli przesyłowych i zasilających do puszek hermetycznych zrealizowano za pomocą złącz przemysłowych producenta WEiPU [36] (rysunek 23). Spełniają one normę szczelności IP68. Oznacza to całkowitą pyłoszczelność i wysoką ochronę przed wodą, pozwalającą nawet na ich zanurzanie.

Rysunek 23: Złącza przemysłowe WEiPU SP13 (Źródło obrazu: strona internetowa producenta [36]).

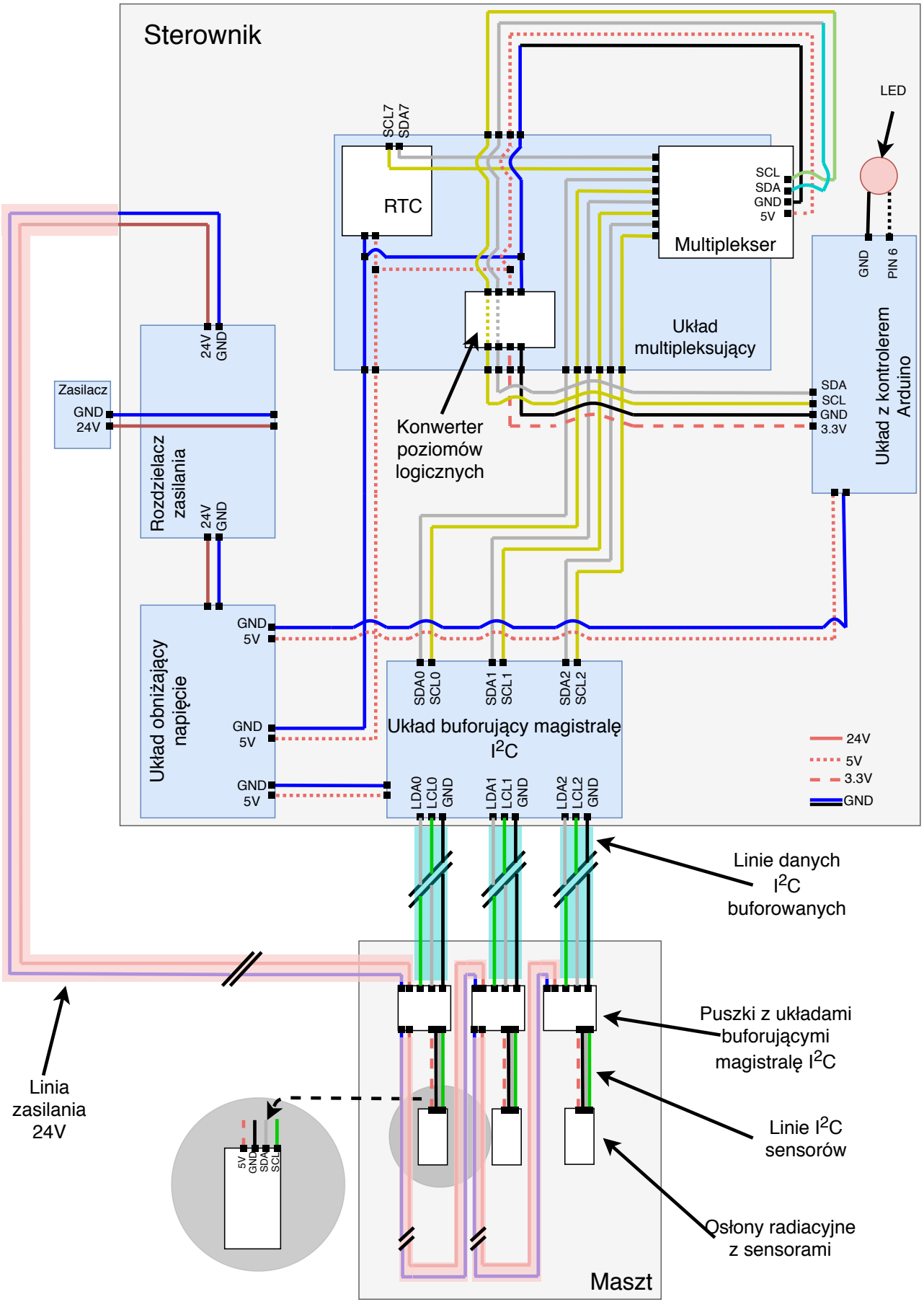
4.4 Dokumentacja techniczna

4.4.1 Schemat blokowy systemu

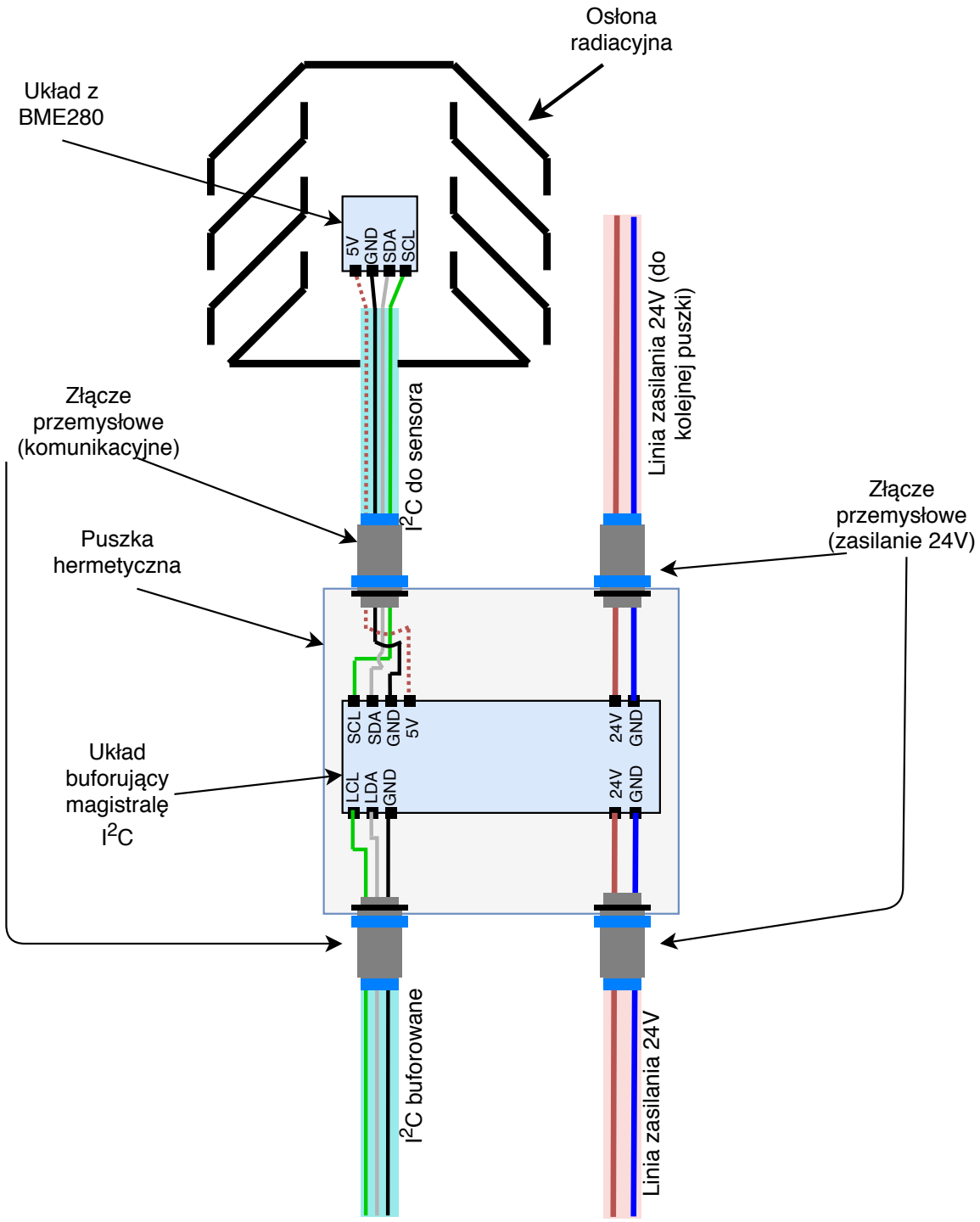
Rysunki 24 - 27 przedstawiają blokowe schematy najważniejszych elementów systemu. Ułożenia komponentów i przewodów oraz kolory przewodów na schematach w przybliżeniu odzwierciedlają ułożenie ich w rzeczywistym systemie. Dokładny opis zasad działania modułów i systemu znajduje się w kolejnych podrozdziałach.



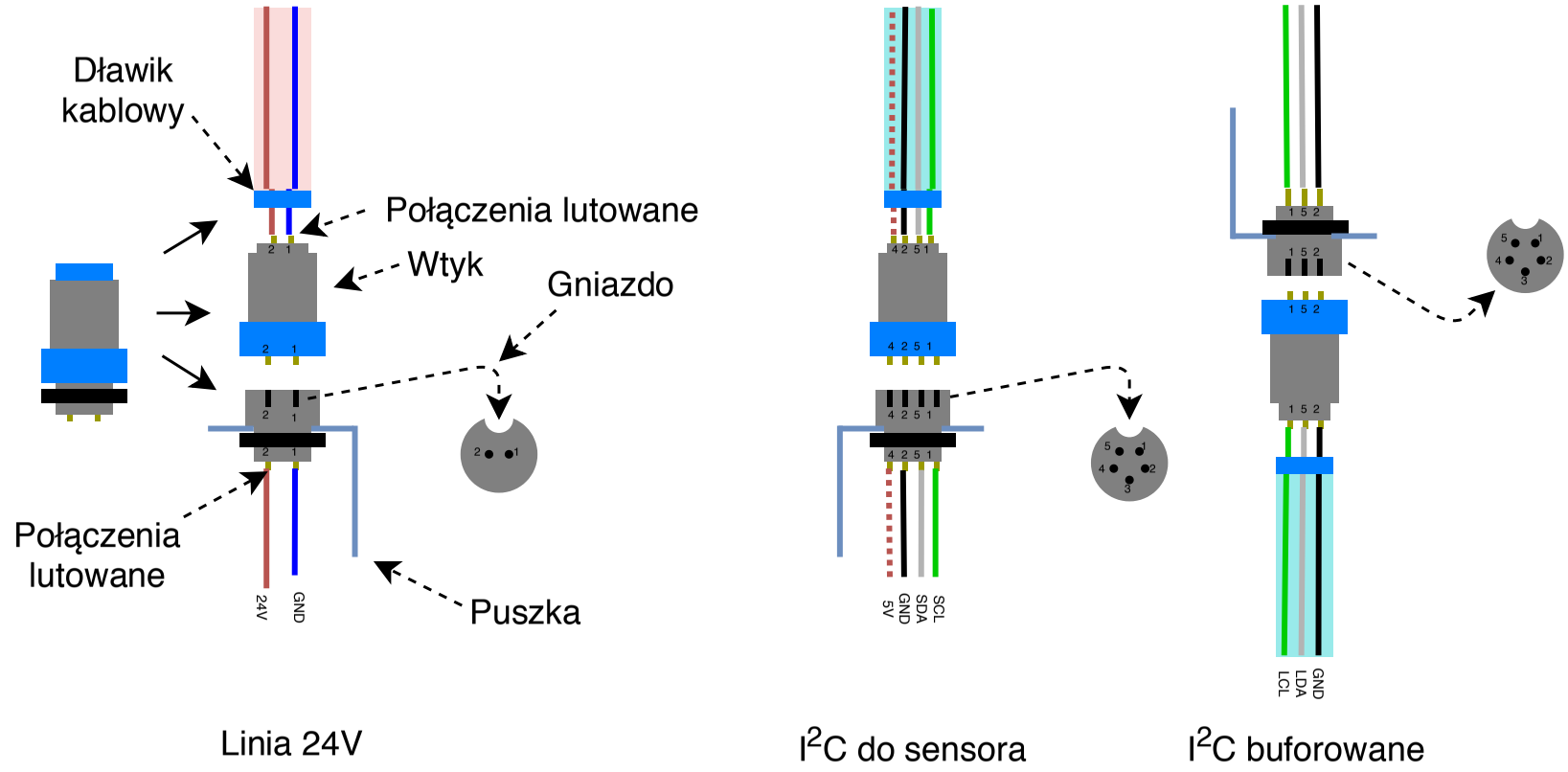
Rysunek 24: Rysunek sytuacyjny układu na maszcie (Skala odległości niezachowana).



Rysunek 25: Schemat blokowy układu. Kolory przewodów i ułożenia elementów w przybliżeniu odzwierciedlają rzeczywisty system.



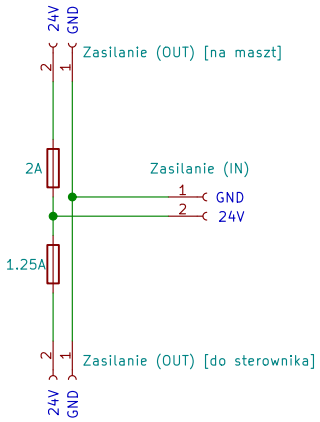
Rysunek 26: Schemat blokowy pojedynczej linii na maszcie. Kolory przewodów i ułożenia elementów w przybliżeniu odzwierciedlają rzeczywisty system.



Rysunek 27: Schemat połączeń złącz przemysłowych. Kolory przewodów i ułożenia elementów w przybliżeniu odzwierciedlają rzeczywisty system

4.4.2 Schematy elektroniczne układów

Zasilanie do układu dostarczane jest przez zasilacz impulsowy 24V. Układ przedstawiony na rysunku 28 rozdziela zasilanie między układ sterownika systemu a układy znajdujące się na maszcie. Zamontowane są w nim bezpieczniki topikowe $1.25A$ do części sterownika i $2A$ do części układów na maszcie.



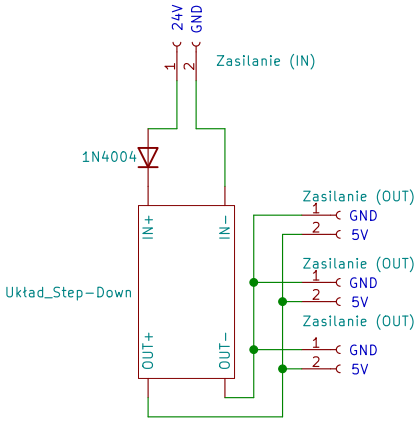
Rysunek 28: Schemat układu rozdzielającego napięcie znajdującego się w sterowniku.

Schemat widoczny na rysunku 29 przedstawia układ obniżający napięcie do 5V. Dioda prostownicza 1N4004 zabezpiecza układ przed przypadkowym podłączeniem odwrotnej polaryzacji zasilania, która mogłoby doprowadzić do jego zniszczenia. Jako układ step-down wykorzystano dostępną w handlu gotową realizację konwertera napięcia opartego o układ LM2596. Jego schemat przedstawia rysunek 30. Sterowanie napięciem wyjściowym opiera się na odpowiednim doborze wartości rezystorów $R1$ i $R2$:

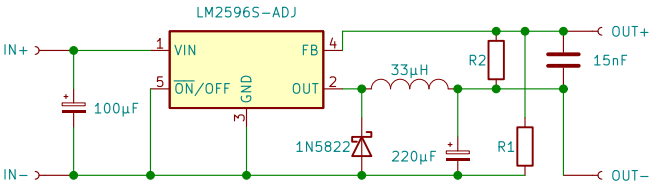
$$V_{out} = 1.23 \cdot \left(1 + \frac{R_2}{R_1}\right) \tag{4.80}$$

W wykorzystywanym układzie rezystor R_1 przyjmuje wartość 330Ω, natomiast R_2 jest potencjometrem regulowanym w zakresie do $10k\Omega$.

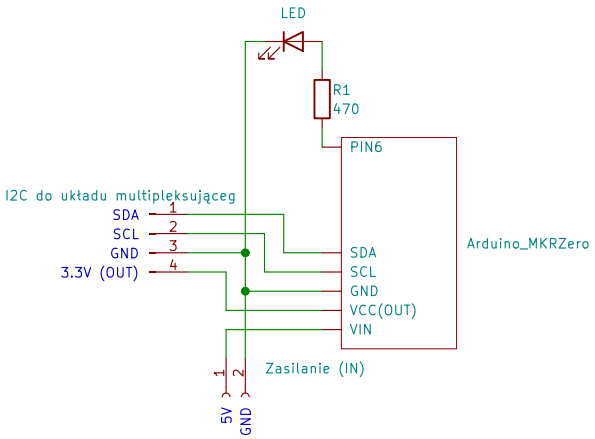
Rysunek 31 przedstawia schemat układu z kontrolerem w postaci Arduino MKRZero. Działa on jako układ *master* w układzie magistrali I^2C i jest połączony z multiplekserem (rysunek 37)



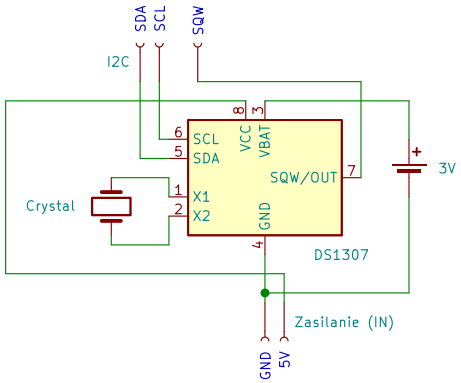
Rysunek 29: Schemat układu obniżającego napięcie znajdującego się w sterowniku.



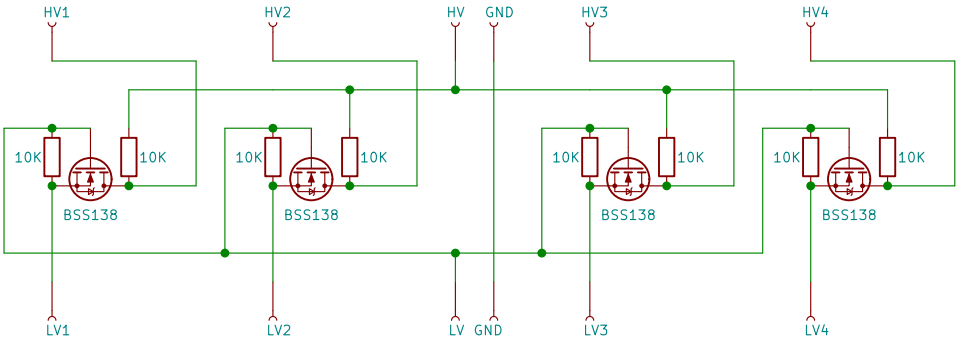
Rysunek 30: Schemat układu Step-Down (Opracowanie własne na podstawie dokumentacji układu LM2596 [12]).



Rysunek 31: Schemat układu z kontrolerem Arduino znajdującego się w sterowniku..



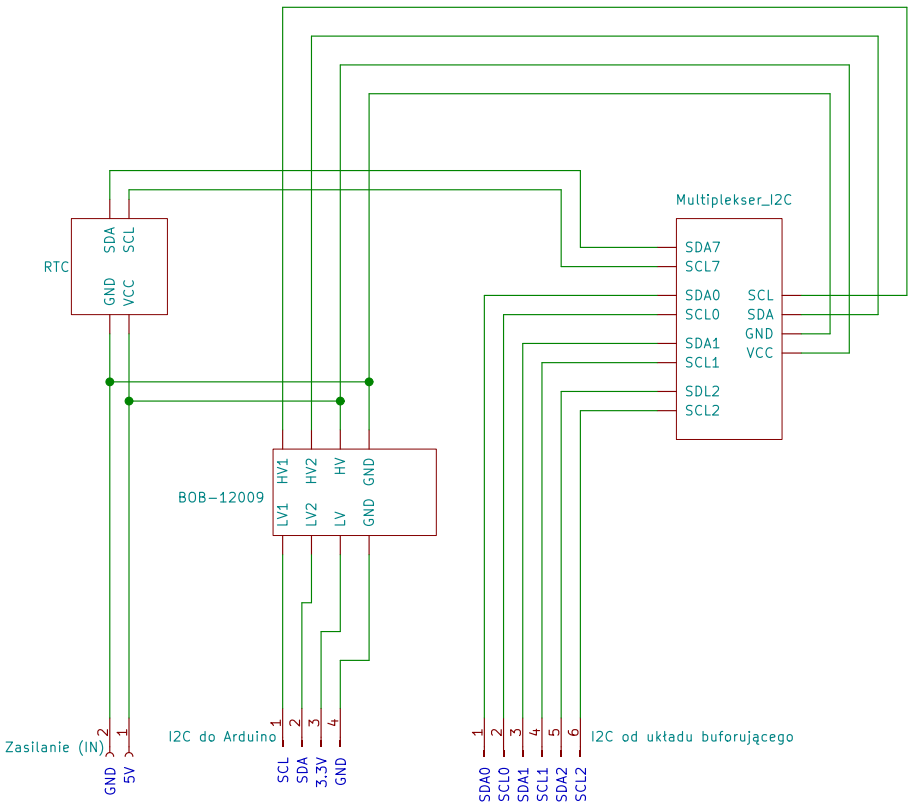
Rysunek 32: Schemat układu zegara czasu rzeczywistego RTC (Opracowanie własne na podstawie dokumentacji układu DS1307 [11]).



Rysunek 33: Schemat układu konwertera poziomów logicznych SparkFun BOB-12009 (Opracowanie własne na podstawie dokumentacji układu BSS138 [10] i informacji na stronie producenta SparkFun BOB-12009 [33]).

poprzez konwerter poziomów logicznych (rysunek 33). Konwerter poziomów logicznych od strony niższego napięcia zasilany jest napięciem Vcc o wartości $3.3V$ pobieranym bezpośrednio z Arduino MKRZero, natomiast napięcie wyższe pobierane jest z przyłączonego zasilania $5V$. Do kanałów multipleksera o numerach od 0 do 2 podłączone są linie biegnące z układu buforującego magistralę I^2C, a do kanału numer 7 podłączony jest zegar czasu rzeczywistego. Rysunek 32 przedstawia schemat zegara czasu rzeczywistego opartego o układ DS1307. Wyprowadzenie SQW stanowi źródło sygnału prostokątnego, które nie jest w projekcie systemu wykorzystywane. W niniejszym systemie użyty został zmontowany już moduł, który dodatkowo został wyposażony w układ pamięci EEPROM. Nie jest ona również wykorzystywana. Układ RTC dodatkowo zasilany jest płaską baterią CR2032, w celu podtrzymania odliczania czasu w momencie odłączenia zasilania.

Jako że zgodnie z informacjami umieszczonymi na stronie producenta, układ multipleksera I^2C jest kompatybilny z poziomami logicznymi o napięciu $5V$, konwerter poziomów logicz-



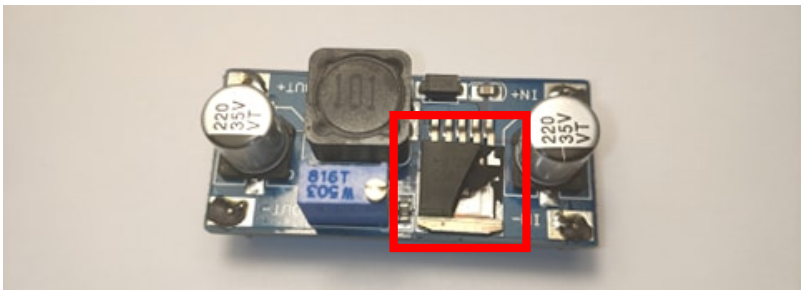
Rysunek 34: Schemat układu multipleksującego magistralę I^2C znajdującego się w sterowniku.

nych postanowiono umieścić między nim, a układem z Arduino, a nie przed każdym wejściem multipleksowanym. Pozwoliło to na zaoszczędzenie miejsca i komponentów.

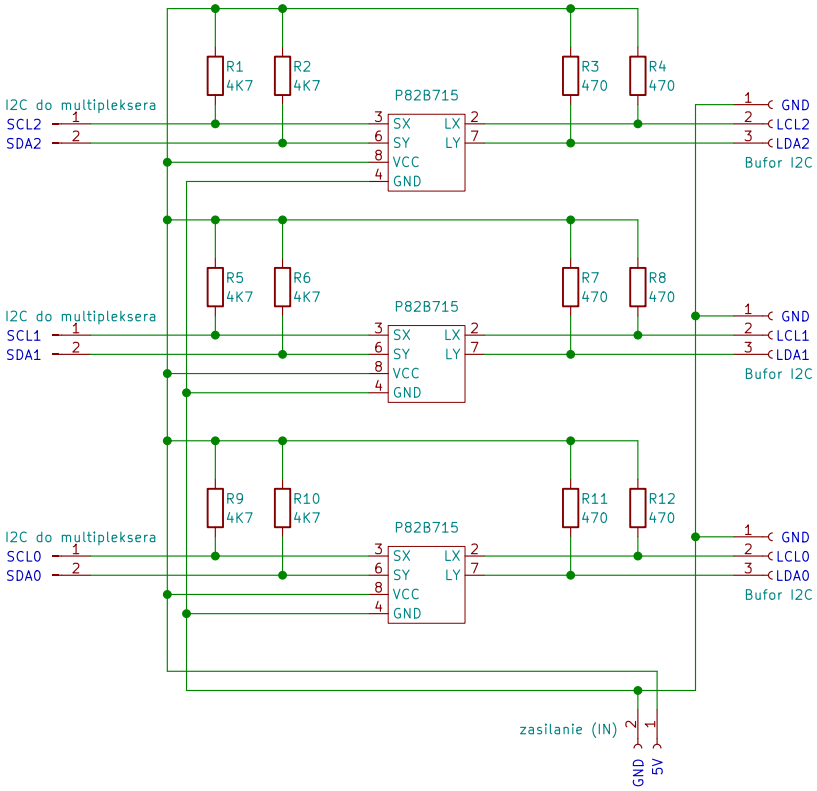
Rysunek 36 przedstawia schemat układu buforującego magistralę I^2C. Składa się od z trzech ułożonych równolegle układów opartych na module P82B715, realizujących przykład typowego zastosowania tego modułu. Bliźniacze układy umieszczone zostały w puszkach instalacyjnych znajdujących się na maszcie przy każdym sensorze (rysunek 38). Dodatkowo znajdują się w nich układy obniżające napięcie step-down, zabezpieczone diodą prostowniczą 1N4004. W układzie tym występuje również bezpiecznik topikowy wmontowany w sposób przedstawiony na ilustracji. Sposób jego zamontowania, w przypadku jego przepalenia, może niestety spowodować przerwanie dostawy zasilania do kolejnych układów. Jest on jednak spowodowany ograniczonym miejscem montażowym w puszce. Oprócz torów LCL i LDA, w linii transmisyjnej występuje również tor GND. Jego obecność spowodowana jest tym, że w transmisjach cyfrowych ważny jest dokładny poziom odniesienia masy układu.

Uwaga!

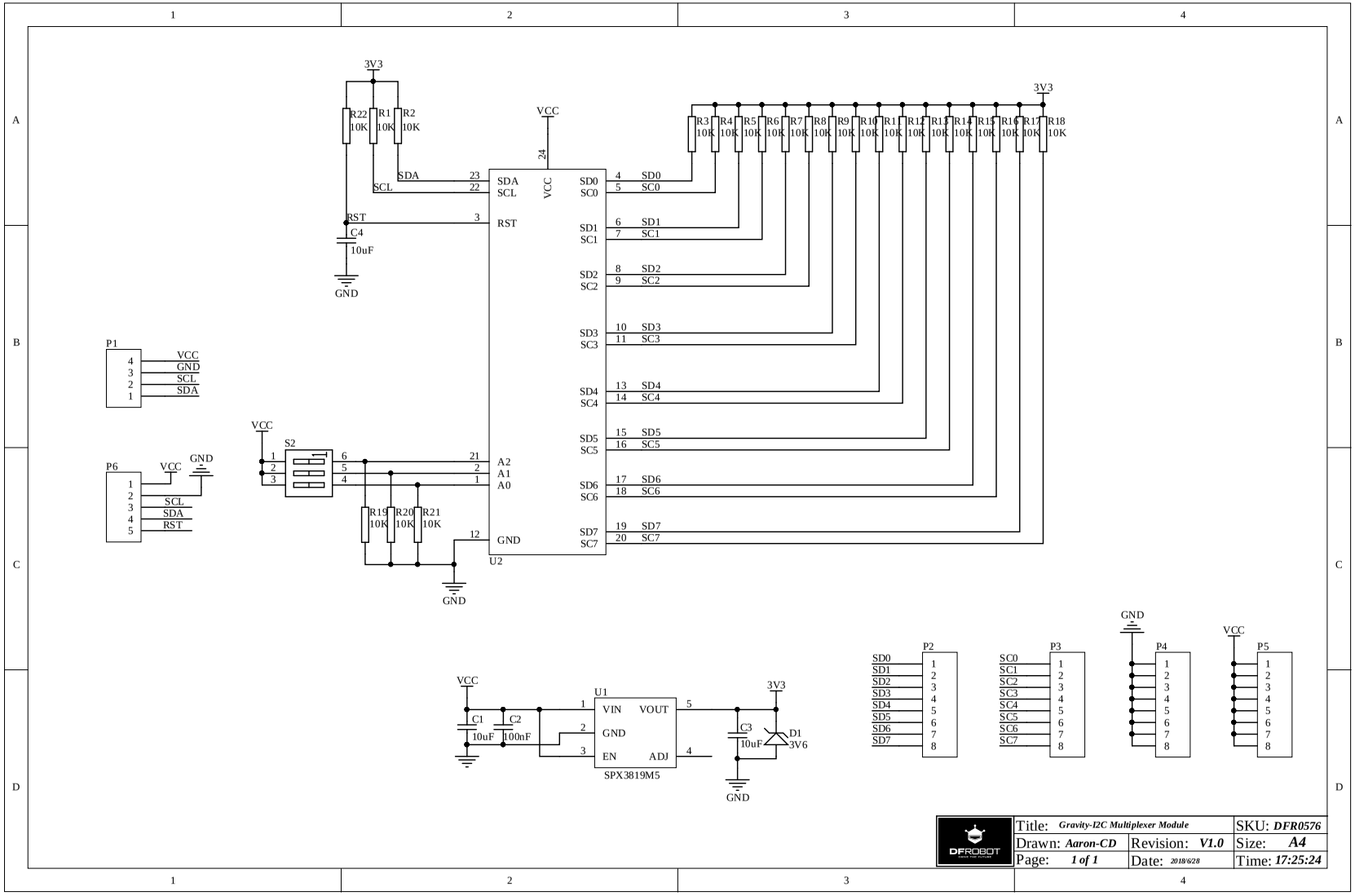
Układy step-down zamontowane w puszkach na maszcie posiadają odwrotną kolejność podłączania zasilania (IN+ i IN-) w stosunku do układu step-down znajdującego się w sterowniku. W przypadku, gdy w przyszłości zajdzie konieczność ich fizycznego wymontowania z układu (na przykład w celu wymiany), należy szczególną uwagę zwrócić na kolejność tych wyprowadzeń, gdyż podłączenie tych układów do odwrotnej polaryzacji napięcia o wartości 24V może doprowadzić do eksplozji kondensatorów elektrolitycznych oraz układu LM2596.



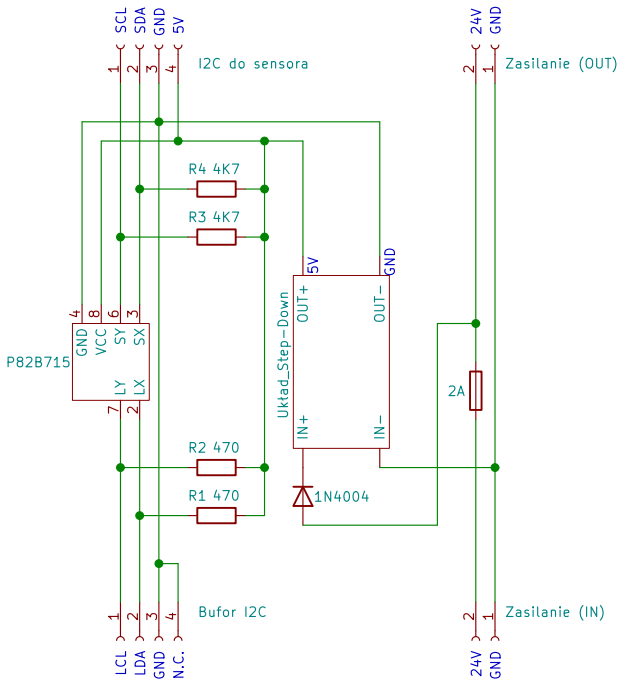
Rysunek 35: Zniszczony układ LM2596 po podłączeniu odwrotnej polaryzacji zasilania.



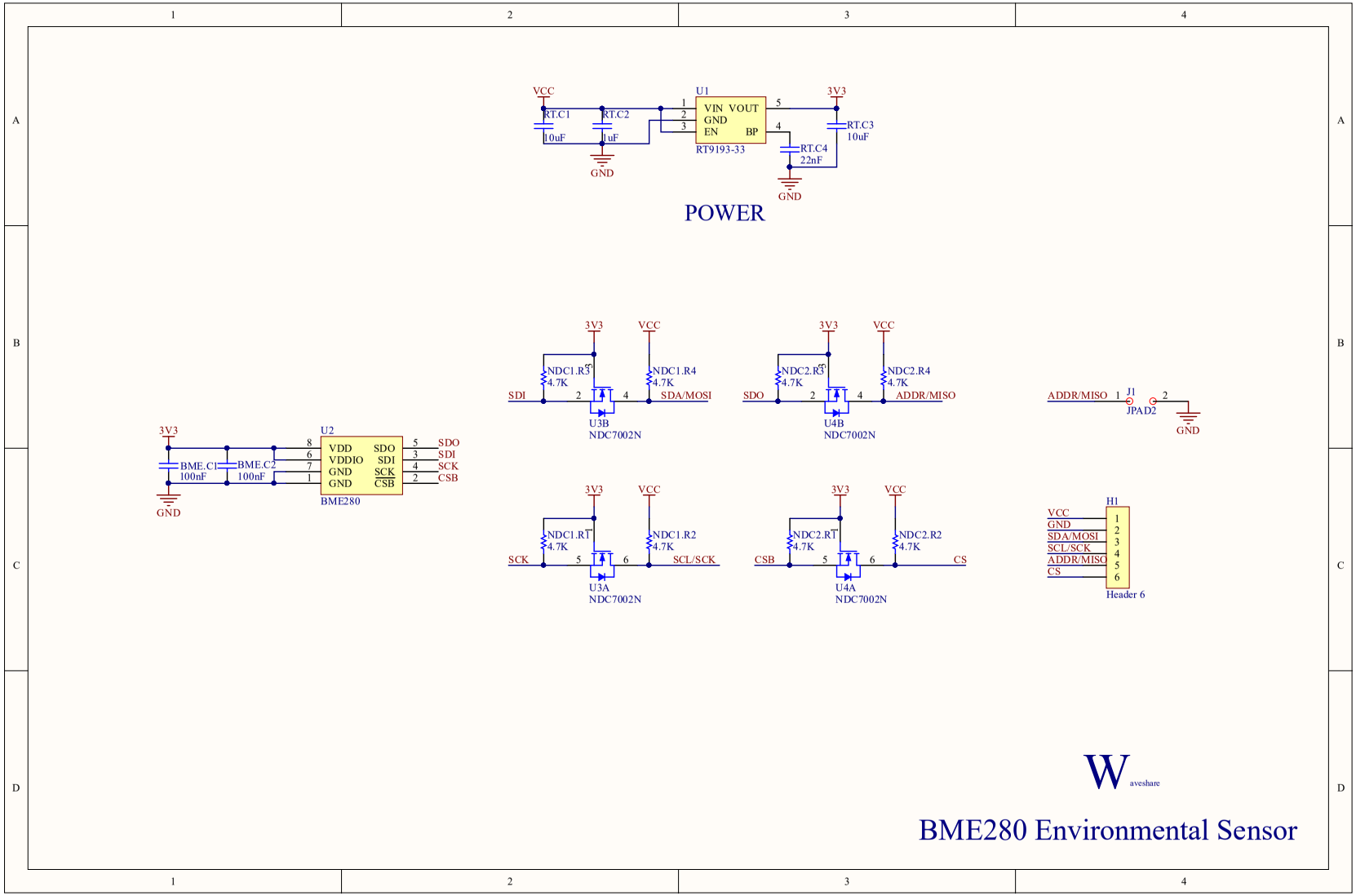
Rysunek 36: Schemat układu buforującego magistralę I^2C znajdującego się w sterowniku.



Rysunek 37: Schemat multipleksera I^2C znajdującego się w układzie multipleksującym. Źródło schematu: strona internetowa producenta [31]



Rysunek 38: Schemat układów buforujących magistralę I^2C znajdujących się w puszkach na maszcie. Układ na szczycie masztu, jako że jest ostatni w szeregu, nie posiada wyprowadzeń zasilania do kolejnego układu.

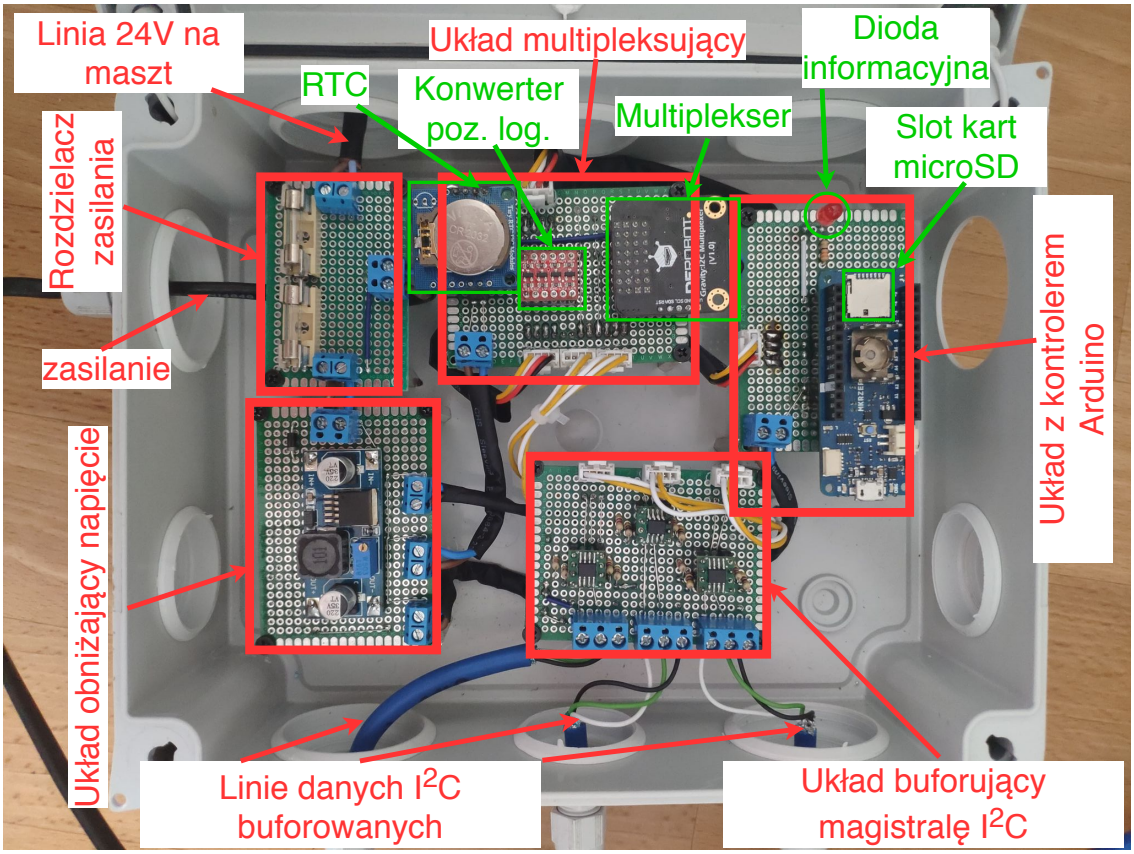


Rysunek 39: Schemat układu sensora opartego na BME280 producenta Waveshare. Źródło schematu: strona internetowa producenta [35]

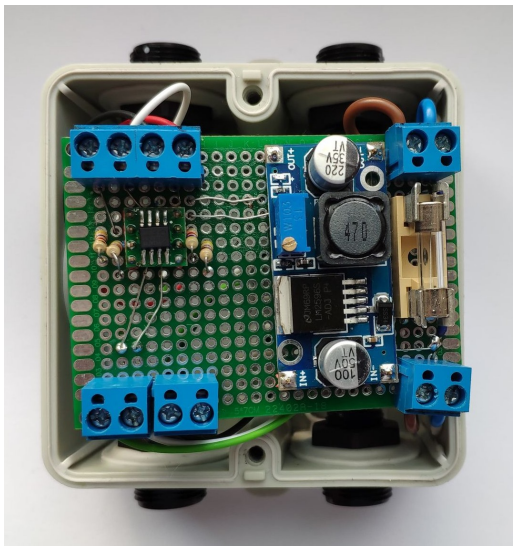
Rysunek 39 przedstawia schemat modułu sensora producenta Waveshare (opartego o układ BME280), wraz z wbudowanymi konwerterami poziomów logicznych (opartymi o układy NDC7002N [13]) oraz stabilizatorem napięcia (opartym o układ RT9193 [15]). Pole JPAD2 pozostaje rozłączone w celu zachowania domyślnego adresu 0x77 dla wszystkich sensorów.

4.4.3 Dokumentacja fotograficzna systemu

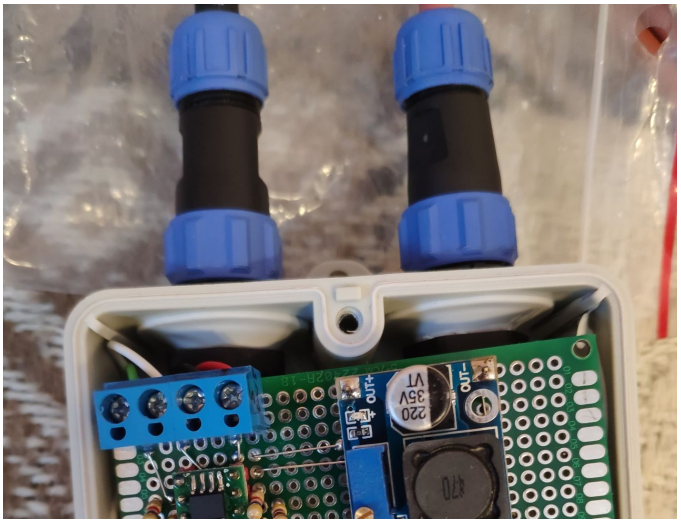
Rysunki 40 - 42 przedstawiają fotografie systemu wykonane w trakcie jego montażu, kalibracji i testowania.



Rysunek 40: Fotografia układu sterownika wraz z opisanymi elementami.



a)



b)

c)

d)

Rysunek 41: a) Układ buforujący magistralę I^2C w puszce, b) Podłączenie złącz przemysłowych, c) Puszka hermetyczna, d) Układ sensorów w osłonach radiacyjnych na maszcie w czasie kalibracji.

Rysunek 42: Fotografia przedstawiająca umiejscowienie sensorów w osłonach radiacyjnych na maszcie[2].

[2]Montaż układu sensorów na maszcie przeprowadził dr inż. Jakub Bartyzel.

4.4.4 Parametry techniczne układu

Tabela 4 przedstawia parametry techniczne zbudowanego systemu. Ze względu na możliwe pojawienie się dużych spadków napięcia na linii zasilającej na maszcie, nie zaleca się zasilania układu napięciem Vcc niższym niż $12V$, a napięciem rekomendowanym jest $Vcc = 24V$. Sterownik systemu powinien być umieszczony w miejscu, w którym nie występują warunki kondensacji pary wodnej.

Sensory BME280 pracują w trybie *forced mode*, z wyłączoną opcją *IIR filter* i parametrem *oversampling* równym 1. Zgodnie z dokumentacją sensora BME280 [9] są to zalecane ustawienia dla pracy w celach meteorologicznych.

Tabela 4: Parametry techniczne systemu do pomiarów strumieni turbulencyjnych ciepła

Ozn.	Nazwa	Wartość min.	Wartość typ.	Wartość max.	Jedn.
N_{BME280}	liczba sensorów BME280 w układzie	-	3	-	szt.
f_{I^2C}	Częstotliwość pracy magistrali I^2C	-	100	-	kHz
h_0	Wysokość sensora 0 nad poziomem dachu	-	≈2.28	-	m
h_1	Wysokość sensora 1 nad poziomem dachu	-	≈10.86	-	m
h_2	Wysokość sensora 2 nad poziomem dachu	-	≈20.00	-	m
Vcc	Napięcie zasilania	12	24	35	V
Icc	Prąd zasilania (przy 24V)	83	95	110	mA
Icc_{ster}	Prąd zasilania sterownika (przy 24V)	65	71	80	mA
Icc_{sens}	Prąd zasilania układu sensora (przy 24V)	6	8	10	mA
T_{ster}	Temperatura pracy sterownika	0	20	50	°C
T_{sens}	Temperatura pracy układu sensorów	-30	20	80	°C
H_{ster}	Wilgotność pracy sterownika	0	-	80	$\%RH$
H_{sens}	Wilgotność pracy układu sensorów	0	-	100	$\%RH$

Tabela 5 przedstawia parametry otoczenia budynku wydziału Fizyki i Informatyki Stosowanej, które mogą być potrzebne w trakcie analiz dotyczących przepływów strumieni turbulencyjnych ciepła w jego obszarze.

Tabela 5: Parametry otoczenia budynku WFiIS

Ozn.	Nazwa	Wartość min.	Wartość typ.	Wartość max.	Jedn.
h_{wiatr}	Wysokość wiatromierza stacji wydziałowej nad poziomem gruntu	-	25.00	-	m
h_{maszt}	Wysokość szczytu masztu nad poziomem gruntu [19]	-	39.70	-	m
h_{WFiIS}	Wysokość budynku WFiIS nad poziomem gruntu	-	19.70	-	m
d	Przesunięcie płaszczyzny zerowej pionowego profilu wiatru [19]	18.25	18.56	18.87	m
z_0	Parametr szorstkości terenu [19]	1.30	1.40	1.50	m

W celu ułatwienia obsługi systemu, oraz detekcji ewentualnych nieprawidłowości jego działania, wprowadzono komunikaty systemowe sygnalizowane przez diodę LED w sterowniku. Ich opis przedstawia tabela 6. Dodatkowo każda aktywność systemu, w postaci włączenia, restartu systemu, detekcji błędnego działania zapisywana jest do pliku *info.LOG* na karcie microSD.

Tabela 6: Komunikaty systemowe

Zachowanie diody	Opis
5-krotne szybkie mignięcie	Początek lub koniec inicjalizacji systemu po włączeniu zasilania.
2-krotne powolne mignięcie	Błąd inicjalizacji karty microSD (na przykład gdy jest nieobecna w porcie).
3-krotne powolne mignięcie	Błąd inicjalizacji zegara czasu rzeczywistego.
4-krotne powolne mignięcie	Błąd inicjalizacji sensora BME280.
5-krotne powolne mignięcie	Niepowodzenie zapisu danych na karcie microSD.
Ciągłe świecenie	Odczytywanie danych z sensorów i ich zapis na kartę SD.
Miganie z częstotliwością 1 raz na sekundę	Okres międzypomiarowy (można wtedy bezpiecznie odłączyć zasilanie i wyciągnąć kartę microSD z portu).

Dane pomiarowe zapisywane są na karcie microSD w postaci plików CSV, których nazwy zawierają aktualną datę (w formie *YYYYMMDD.CSV*). Czas zapisywany jest w strefie czasowej UTC. Przykładowy fragment pliku znajduje się w dodatku A.

4.4.5 Dokumentacja programu sterownika

Niniejsza sekcja przedstawia opis zmiennych i efektów działania poszczególnych funkcji programu sterownika układu. Cały jego kod znajduje się w dodatku B.

int countDown;

Opis: *Zmienna do której przypisano wartość zwracaną przez konstruktor klasy obsługującej watchdog, którego zegar ustawiono na 30 sekund.*

RTC_DS1307 rtc;

Opis: *Instancja klasy obsługującej zegar czasu rzeczywistego.*

Adafruit_BME280 bme0; Adafruit_BME280 bme1; Adafruit_BME280 bme2;

Opis: *Instancje klas obsługujących sensory BME280.*

Adafruit_BME280 BME[3];

Opis: *Tablica przechowująca instancje klas obsługujących sensory BME280.*

bool sensorCheck[3];

Opis: *Tablica przechowująca informacje o poprawności inicjalizacji poszczególnych sensorów BME280.*

uint8_t I2CMultiplexer;

Opis: *Adres multipleksera I^2C.*

uint8_t RTCPort;

Opis: *Numer kanału multipleksera do którego podłączony jest zegar czasu rzeczywistego.*

uint8_t sensor0; uint8_t sensor1; uint8_t sensor2;

Opis: *Numery kanałów multipleksera do których podłączone są układy sensorów BME280.*

uint8_t sensorMin; uint8_t sensorMax;

Opis: *Najmniejszy i największy numer kanału multipleksera do którego podłączone są układy sensorów BME280.*

uint8_t LEDPin;

Opis: *Numer pinu Arduino MKRZero do którego podłączona jest dioda sygnalizacyjna.*

const int chipSelect;

Opis: *Numer linii czytnika kart microSD w układzie kontrolera Arduino MKRZero.*

const int resetCounter;

Opis: *Wartość ilości iteracji pętli głównej programu po której ma nastąpić ponowna inicjalizacja systemu.*

int counter;

Opis: *Licznik ilości iteracji pętli głównej programu sterownika systemu.*

float sensorTemperatureCalibration[3][2];

Opis: *Tablica zawierające współczynniki kalibracyjne temperatury odczytanej z sensorów BME280.*

float sensorHumidityCalibration[3][2];

Opis: *Tablica zawierające współczynniki kalibracyjne wilgotności odczytanej z sensorów BME280.*

float sensorPressureCalibration[3][2];

Opis: *Tablica zawierające współczynniki kalibracyjne ciśnienia odczytanego z sensorów BME280.*

void selectPort(uint8_t port);

Parametry: port - *numer kanału multipleksera z zakresu od 0 do 7*

Opis: *Wybiera kanał multipleksera I^2C w celu późniejszej komunikacji z odpowiednim podzespołem. Implementacja funkcji zaczerpnięta z biblioteki producenta multipleksera I^2C [2], gdyż importowanie kodu całej biblioteki zawiesza Arduino MKRZero.*

String getSensorValues(uint8_t port);

Parametry: port - *numer kanału multipleksera do którego wpięty jest konkretny sensor.*

Opis: *Zwraca w formacie ciągu znaków odczyty z sensorów BME280 podłączonych do wybranego kanału multipleksera w formie: "T,T_cal,H,H_cal,P,P_cal", (gdzie: T - temperatura, H - wilgotność, P - ciśnienie, X_cal - wartość odpowiedniej wielkości po kalibracji) lub w przypadku błędu odczytu którejkolwiek wartości: "nan,nan,nan,nan,nan,nan".*

String getTime();

Opis: *Zwraca w formacie ciągu znaków odczyt aktualnego czasu w formie "HH:MM:SS" (gdzie: HH - godzina, MM - minuta, SS - sekunda).*

uint8_t getHour();

Opis: *Zwraca w formacie uint8_t odczyt aktualnej godziny.*

uint8_t getMinute();

Opis: *Zwraca w formacie uint8_t odczyt aktualnej minuty.*

uint8_t getSecond();

Opis: *Zwraca w formacie uint8_t odczyt aktualnej sekundy.*

String getDate();

Opis: *Zwraca w formacie ciągu znaków odczyt aktualnej daty w formie "YYYYMMDD" (gdzie: YYYY - rok, MM - miesiąc, DD - dzień).*

void writeData(String dataString, String filename)

Parametry: dataString - *tekst do zapisu na karcie microSD*, filename - *nazwa pliku*

Opis: *Zapisuje na karcie microSD w pliku o nazwie "filename" linijkę tekstu ze zmiennej "dataString". Gdy plik o podanej nazwie nie istnieje, to zostaje utworzony.*

void flashLED(int number, int len);

Parametry: number - *ilość mignięć*, len - *długość mignięcia*

Opis: *Powoduje miganie diody informacyjnej sterownika określoną ilość razy przy określonej długości trwania jednego mignięcia.*

void initializeSDCard();

Opis: *Inicjalizacja czytnika kart microSD i biblioteki do jego obsługi.*

void initializeI2CMultiplexer();

Opis: *Inicjalizacja multipleksera I^2C.*

void initializeRTC();

Opis: *Inicjalizacja zegara czasu rzeczywistego RTC i biblioteki do jego obsługi. Jako że poprawna inicjalizacja RTC jest niezbędna do poprawnego działania systemu, to w razie jej niepowodzenia w nieskończoność następują jej kolejne próby. Niepowodzenie inicjalizacji sygnalizowane jest potrójnym powolnym mignięciem diody sygnalizacyjnej.*

void initializeSensors();

Opis: *Inicjalizacja sensorów BME280 i bibliotek do ich obsługi. Dla każdego sensora następuje pięciokrotna próba jego inicjalizacji. Niepowodzenie próby inicjalizacji sygnalizowane jest poczwórnym powolnym mignięciem diody sygnalizacyjnej oraz zapisaniem komunikatu błędu wraz z numerem sensora i aktualnym czasem do pliku "info.LOG" na karcie microSD. Ponadto informacja o błędnej inicjaizacji zapisywana jest w tablicy sensorCheck[], w celu pominięcia odczytu z sensora w pętli głównej programu.*

void initializeAll();

Opis: *Funkcja wywołuje funkcje inicjalizacyjne sensorów, zegara czasu rzeczywistego, multipleksera i czytnika kart microSD.*

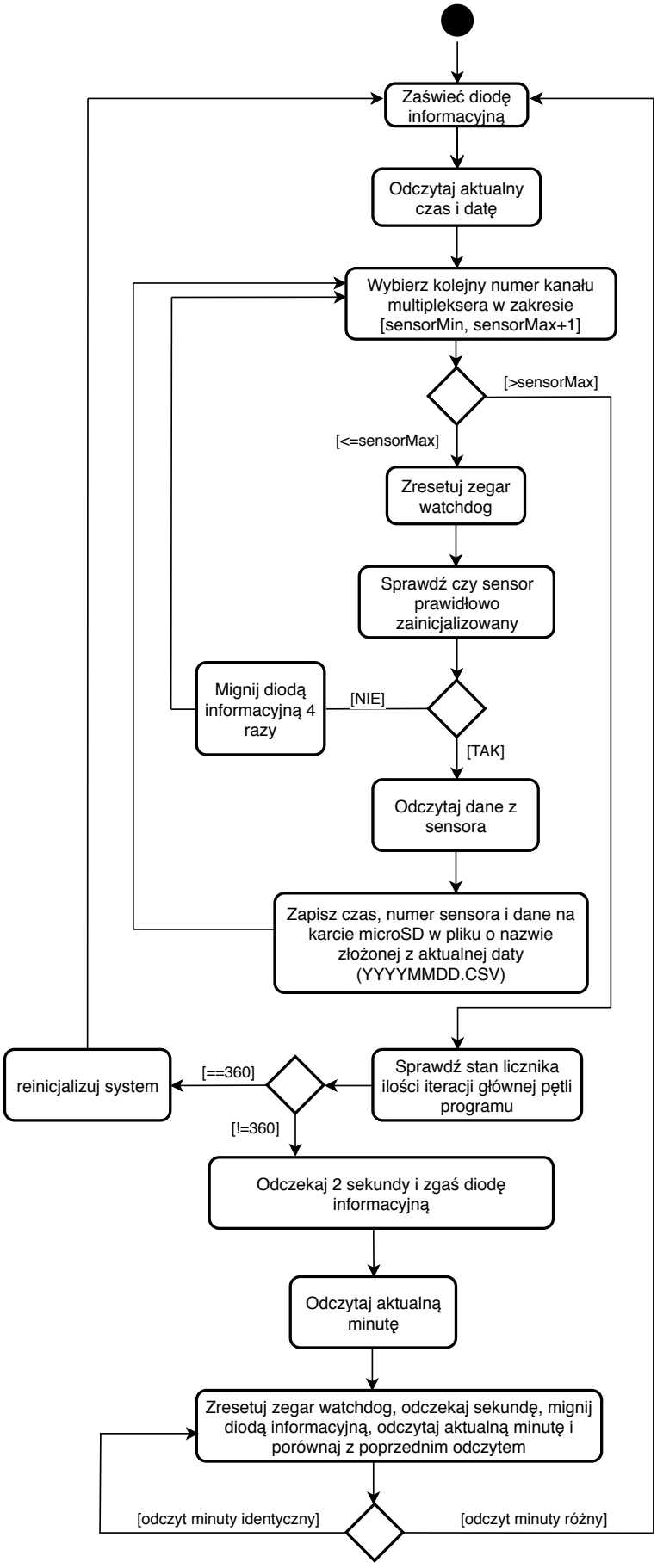
void setup();

Opis: *Funkcja wywołująca się w momencie włączenia kontrolera Arduino MKRZero. Wywołuje funkcję initializeAll oraz inicjalizuje diodę informacyjną systemu.*

void loop();

Opis: *Główna pętla sterownika systemu. Schemat jej działania (a zarazem schemat działania całego systemu) przedstawia rysunek 43.*

Wielkość sensor_max na diagramie na rysynku 43 odpowiada liczbie sensorów w systemie liczonych od zera (czyli liczbie 2). Ilość 360 iteracji głównej pętli programu odpowiada 6 godzinom pracy. Reinicjalizacja systemu po tym czasie została wprowadzona w celu minimalizacji ryzyka błędnych odczytów z sensorów pojawiających się w czasie ich długiej pracy. Obsługa zegara watchdog'a wbudowanego w procesor Arduino MKRZero została zrealizowana za pomocą biblioteki Adafruit SleepyDog [3]. Jeśli jego zegar nie będzie resetowany co 30 sekund, na przykład w wyniku zawieszenia się systemu, watchdog doprowadzi do restartu kontrolera Arduino MKRZero (a więc do restartu całego systemu).



Rysunek 43: Diagram aktywności głównej pętli programu sterownika.

4.4.6 Kalibracja układu

Przed właściwym zamontowaniem sensorów na maszcie, dokonano ich kalibracji względem stacji pogodowej Davis Instruments Vantage Pro [30]. Pomiary kalibracyjne prowadzono w dniach 4 września - 8 września 2020 roku. Niestety, z powodu nieprawidłowego działania stacji, nie udało się uzyskać danych w pełnym zakresie czasowym (rysunek 44). Zebrane dane pozwoliły jednak na dokonanie poprawnej kalibracji, która polegała na dopasowaniu metodą najmniejszych kwadratów krzywej o równaniu:

$$y = a \cdot x + b \tag{4.81}$$

do zależności między wartościami pomiarów dokonanych przez stację Davis, a wartościami pomiarów dokonanych przez sensory BME280.

Jako miarę jakości dopasowania krzywych kalibracyjnych wykorzystano następujące wielkości:

- Współczynnik korelacji Pearson'a:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.82}$$

  Przyjmuje on wartości z zakresu [-1, 1]. Im jego bezwzględna wartość jest bliższa jedynki, tym między zmiennymi występuje większa zależność liniowa.

- Standardowy błąd dopasowania zbocza:

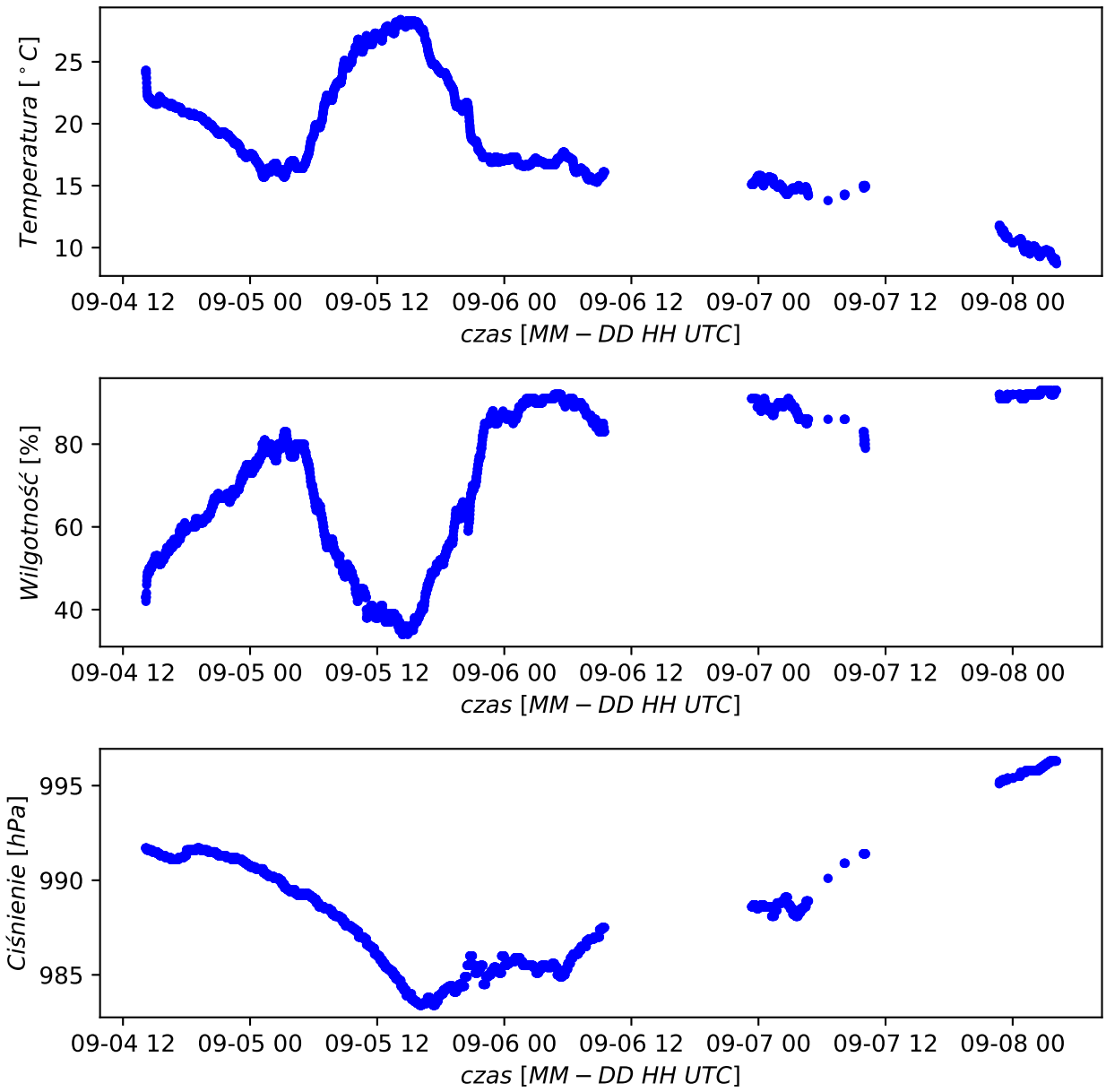
$$\sigma_a = \sqrt{\frac{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \widehat{y})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}} \tag{4.83}$$

- Standardowy błąd dopasowania reszty:

$$\sigma_b = \sqrt{\frac{\sum_{i=i}^{n}(y_i - \widehat{y})^2}{n-2}} \tag{4.84}$$

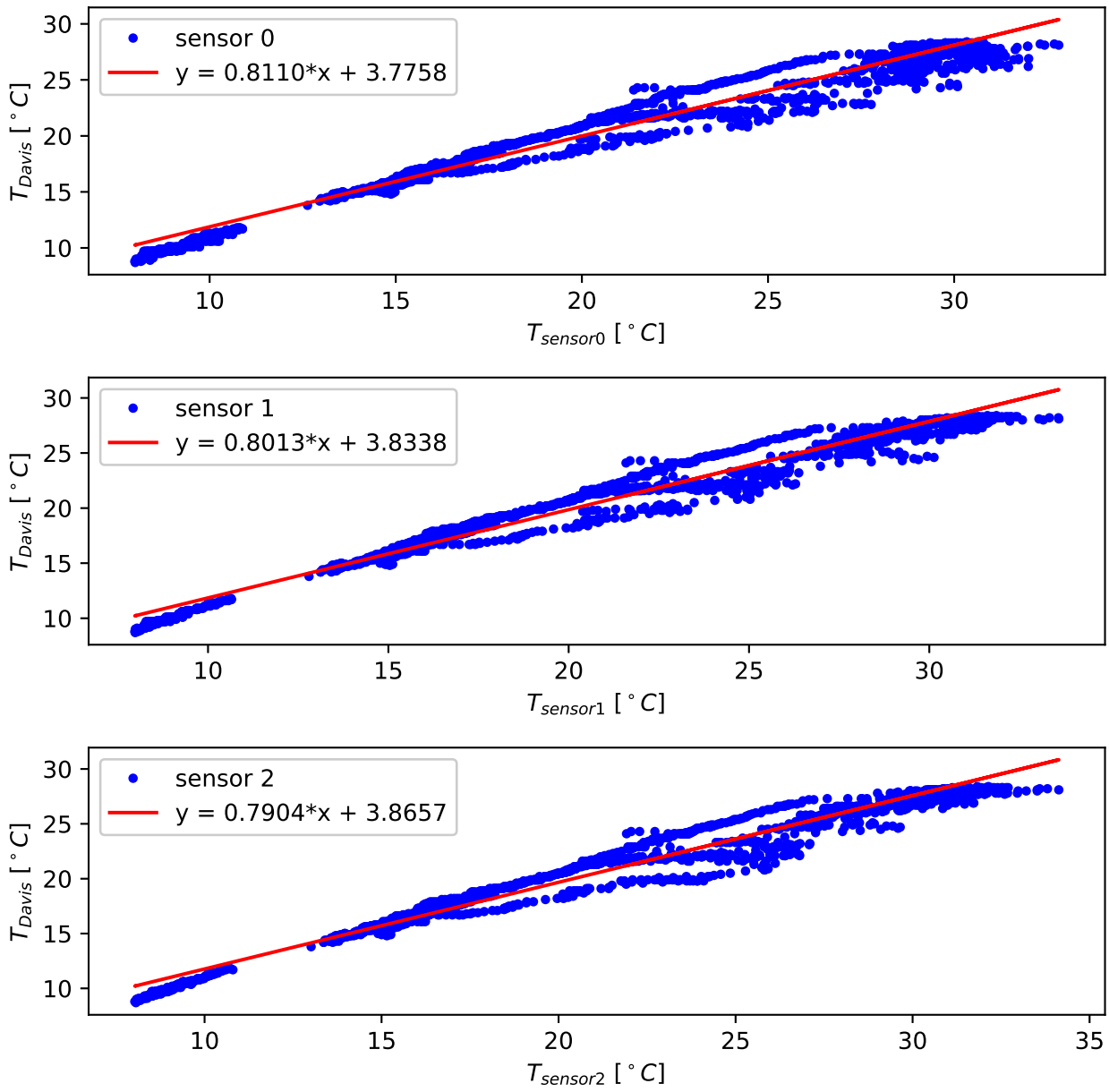
  Określa jaka jest przeciętna różnica między wartościami referencyjnymi, a tymi wyznaczonymi z modelu regresji.

Do przeprowadzenia analizy wykorzystano język Python [7] oraz biblioteki Pandas [5] i NumPy [4]. Kod skryptu kalibracyjnego znajduje się w dodatku C.



Rysunek 44: Odczyty parametrów meteorologicznych stacji Davis wykorzystane do kalibracji sensorów.

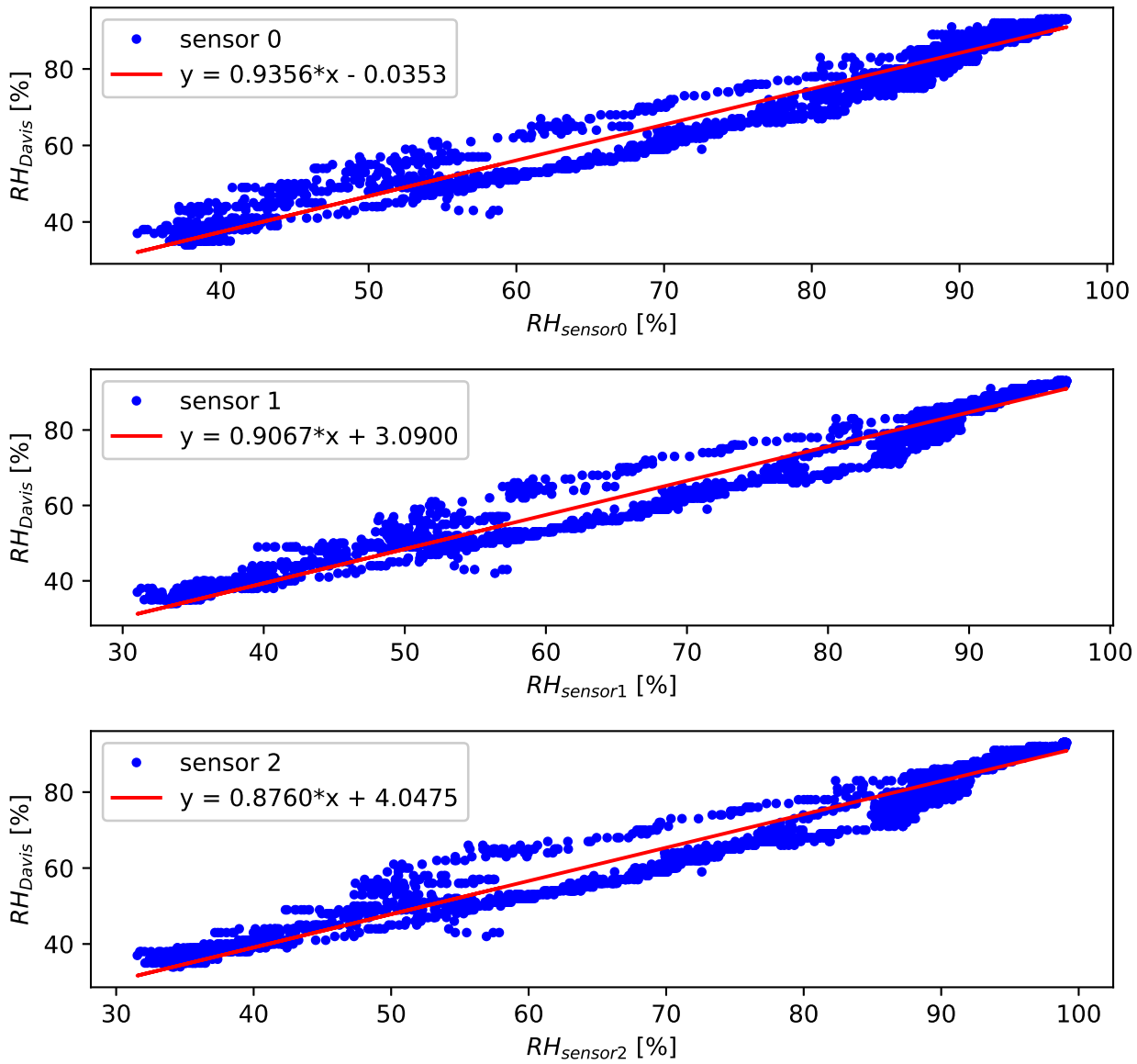
Duży rozrzut wartości odczytywanych temperatur w zakresie $17°C - 30°C$ (rysunek 45) i wilgotności (rysunek 46) spowodowany jest różnymi bezwładnościami temperaturowymi i wilgotnościowymi stacji Davis oraz kalibrowanych sensorów BME280.



Rysunek 45: Wykres krzywych kalibracyjnych sensorów BME280 dla temperatury względem stacji Davis.

Tabela 7: Parametry kalibracji sensorów BME280 dla temperatury

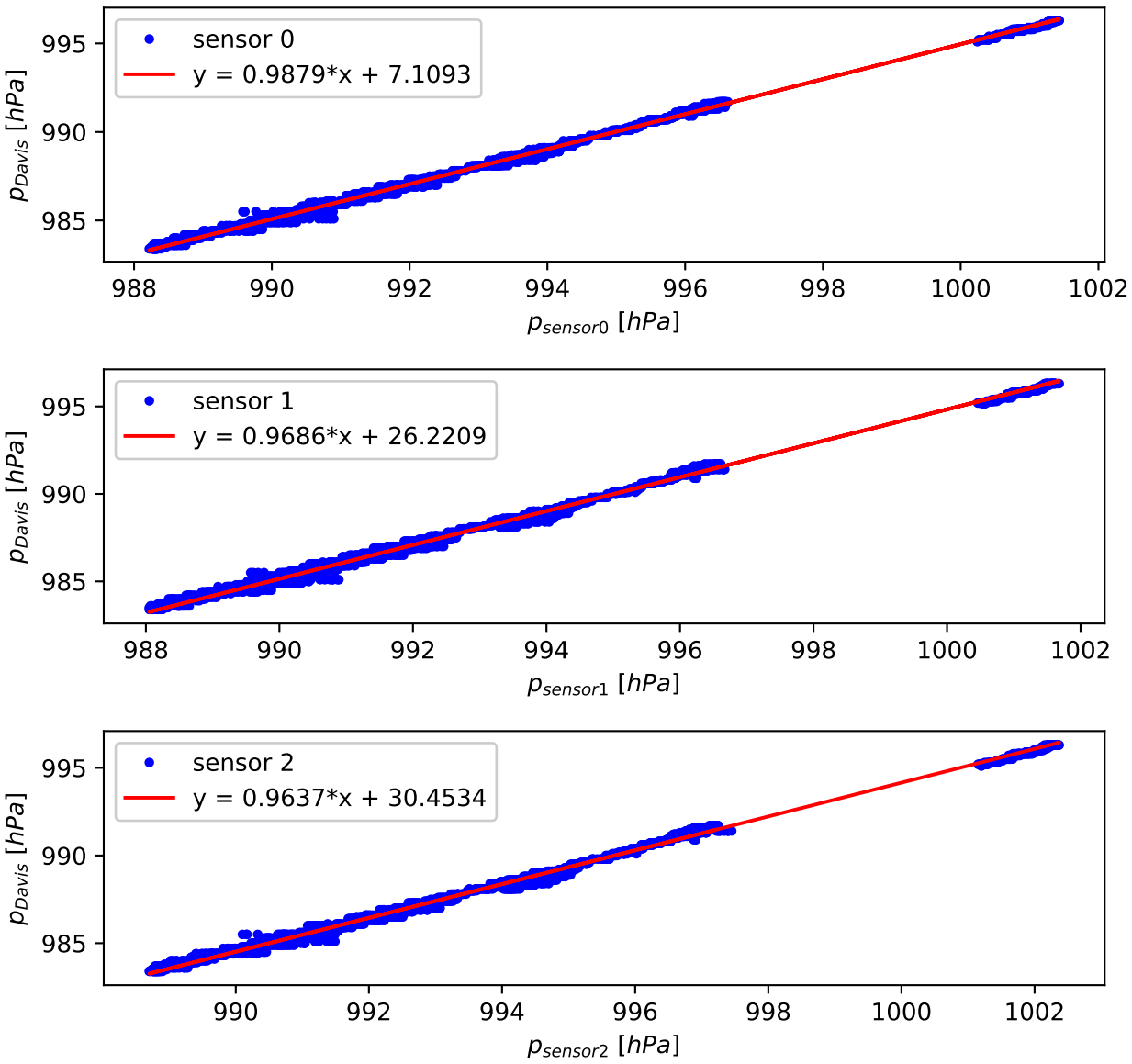
sensor	współczynnik a	współczynnik b	współczynnik r	σ_a	σ_b
0	0.8110	3.7758	0.9826	0.0027	0.0533
1	0.8013	3.8338	0.9848	0.0025	0.0496
2	0.7904	3.8657	0.9836	0.0026	0.0514



Rysunek 46: Wykres krzywych kalibracyjnych sensorów BME280 dla wilgotności względem stacji Davis.

Tabela 8: Parametry kalibracji sensorów BME280 dla wilgotności

sensor	współczynnik a	współczynnik b	współczynnik r	σ_a	σ_b
0	0.9356	-0.0353	0.9796	0.0034	0.2664
1	0.9067	3.0900	0.9800	0.0033	0.2529
2	0.7904	4.0475	0.9794	0.0032	0.2528



Rysunek 47: Wykres krzywych kalibracyjnych sensorów BME280 dla ciśnienia względem stacji Davis.

Tabela 9: Parametry kalibracji sensorów BME280 dla ciśnienia

sensor	współczynnik a	współczynnik b	współczynnik r	σ_a	σ_b
0	0.9879	7.1093	0.9991	0.0008	0.7552
1	0.9686	26.2209	0.9983	0.0010	0.9888
2	0.9637	30.4534	0.9983	0.0010	0.9953

Po wyznaczeniu krzywych kalibracyjnych do sterownika została wgrana poprawka, uwzględniająca powyższe wyliczenia. Sterownik, w celu zapewnienia możliwości pracy na oryginalnych

danych przed poprawką, zapisuje też dane nieskalibrowane.

5 Testowanie systemu

W dniach od 11 września 2020 roku do 21 września 2020 roku przeprowadzono testowe pomiary z wykorzystaniem układu umieszczonego na maszcie. Zebrane przez układ dane w postaci skalibrowanych wartości temperatur, wilgotności i ciśnienia na trzech wysokościach uśredniono w przedziałach 30-minutowych. Podobnie postąpiono z danymi dotyczącymi poziomej prędkości wiatru, pochodzącymi z wydziałowej stacji meteorologicznej. Następnie, dokonano przeliczenia temperatur wyrażonych w $^{\circ}C$ na temperatury absolutne wyrażone w Kelwinach oraz wilgotności względnej, wyrażonej w punktach procentowych, na wilgotność właściwą, wyrażoną w jednostce $kg \cdot kg^{-1}$. Algorytm przeliczeniowy wilgotności przedstawiają poniższe równania [23]:

$$e_s(T) = \begin{cases} exp\left(\frac{-6141}{T} + 24.3\right), & dla\ T < 273.15K \\ exp\left(\frac{-6763.6}{T} - 4.9283 \cdot lnT + 54.23\right), & dla\ T \geq 273.15K \end{cases} \tag{5.85}$$

gdzie: e_s - ciśnienie pary nasyconej,

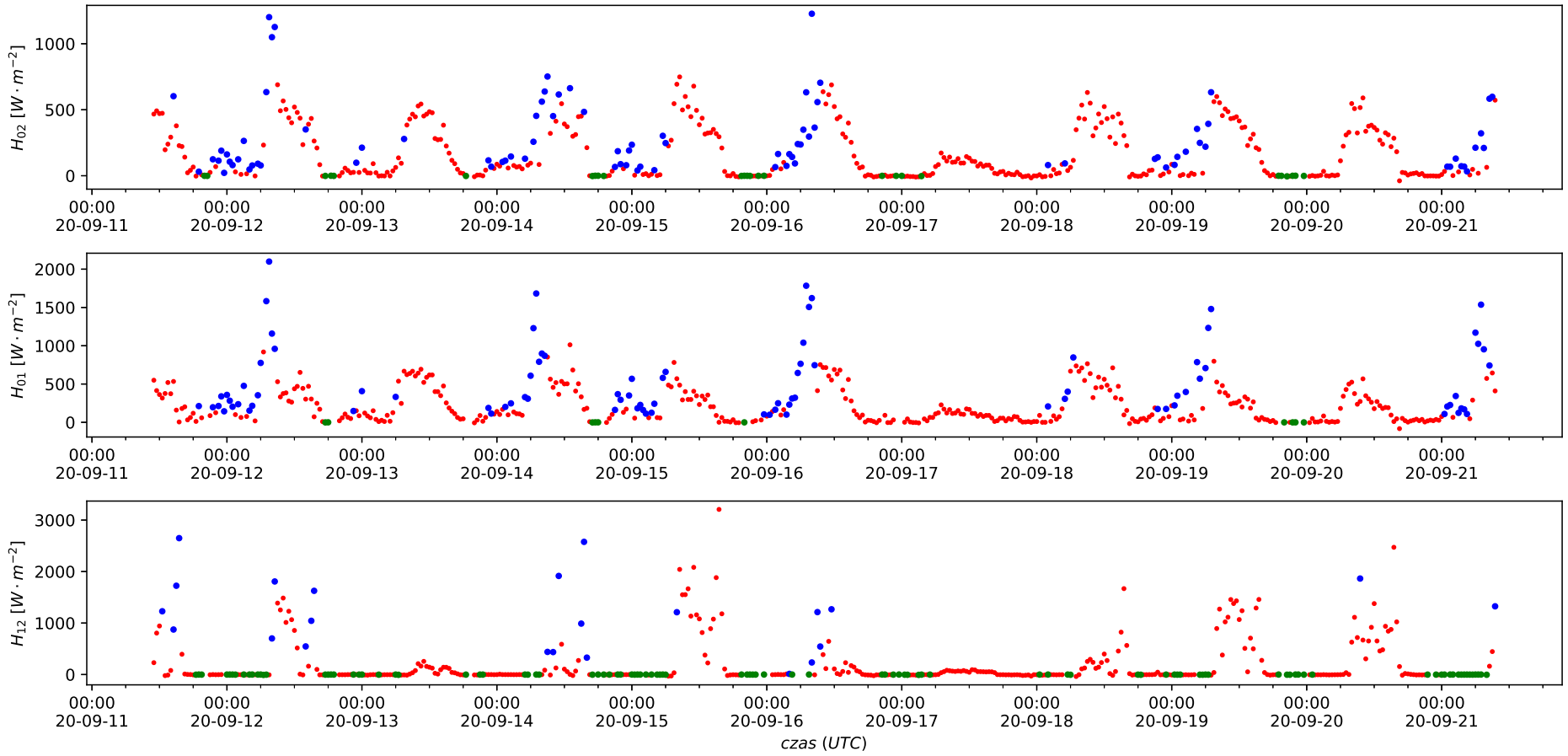
$$e = e_s(T) \cdot \frac{RH}{100} \tag{5.86}$$

gdzie: e - ciśnienie pary w powietrzu o wilgotności względnej RH (wyrażonej w %),

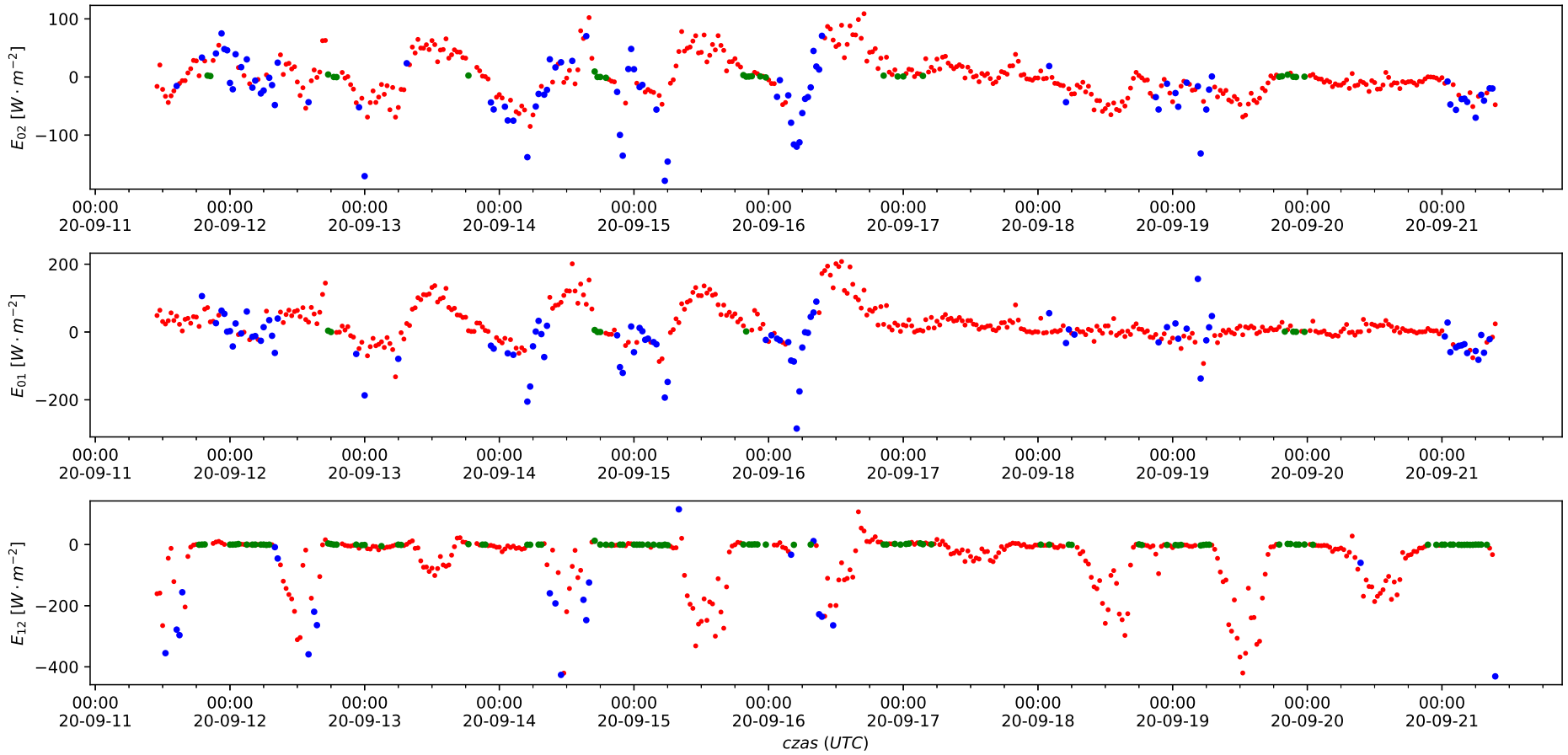
$$q = \frac{0.622 \cdot e}{p - 0.378 \cdot e} \tag{5.87}$$

gdzie: q - wilgotność właściwa, $[kg \cdot kg^{-1}]$; p - ciśnienie powietrza, $[hPa]$.

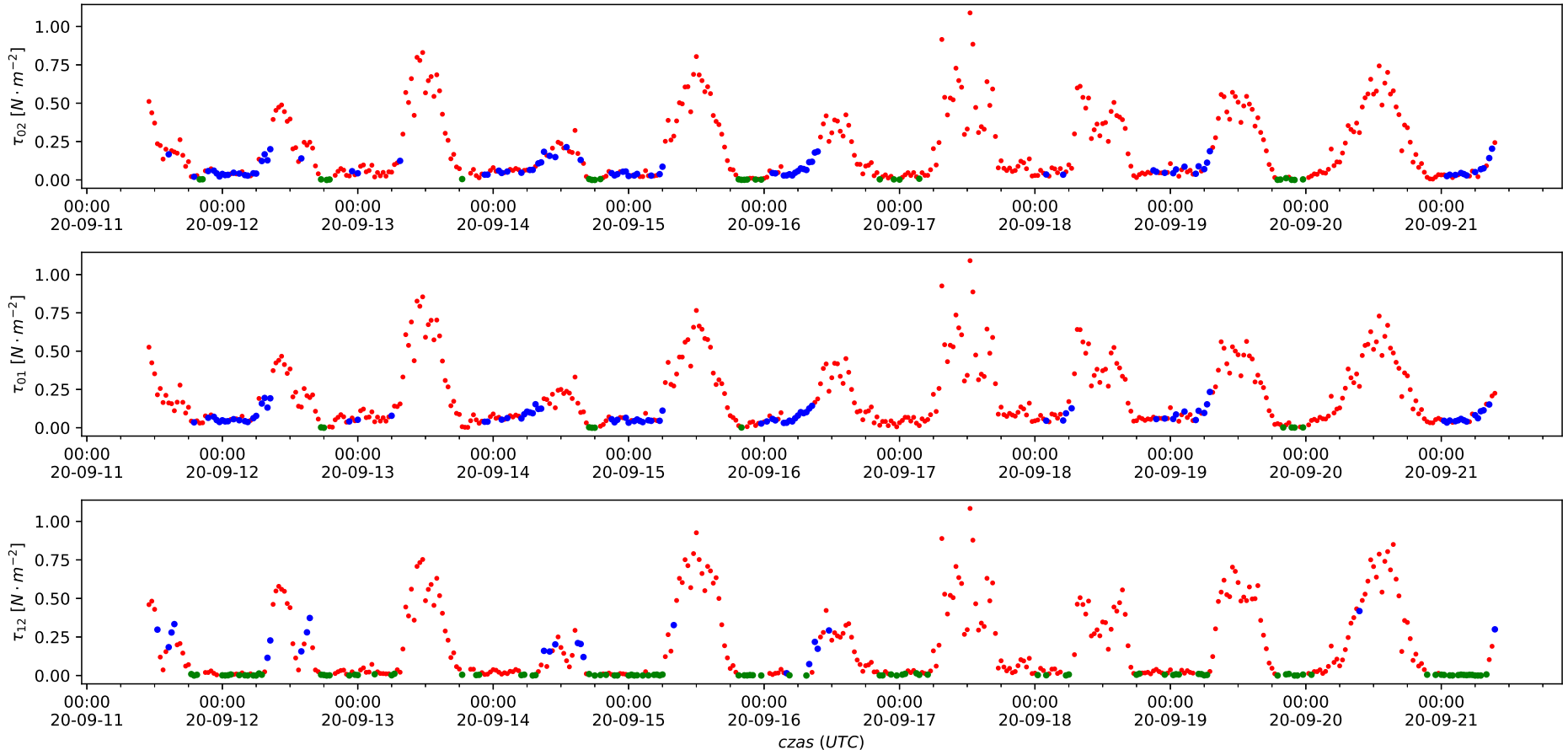
Następnie, na podstawie metody gradientowej opartej na metodzie bulk transfer (rysunek 5 w rozdziale *Metodyka pomiarowa turbulencyjnych strumieni ciepła*), obliczono strumienie turbulencyjne ciepła jawnego H, utajonego E, pędu τ oraz prędkość tarciową u_* i współczynniki ζ między każdymi poziomami sensorów na maszcie. Obliczenia przeprowadzono z wykorzystaniem języka Python i bibliotek Pandas i NumPy. Kod skryptu obliczeniowego znajduje się w dodatku D. Jako parametry otoczenia wydziału w postaci przesunięcia płaszczyzny zerowej profilu wiatru d, szorstkości terenu z_0 i wysokości wiatromierza nad poziomem gruntu wykorzystano dane znajdujące się w tabeli 5 w rozdziale *Dokumentacja techniczna*. Jako ciepło właściwe powietrza przyjęto wartość $c_p = 1005\ J \cdot kg^{-1} \cdot K^{-1}$, jako ciepło właściwe parowania wartość $l = 2257000\ J \cdot kg^{-1} \cdot K^{-1}$ i jako gęstość powietrza $\rho = 1.25\ kg \cdot m^{-3}$. Rysunki 48 - 52 przedstawiają wyniki obliczeń strumieni na podstawie wykonanych pomiarów.



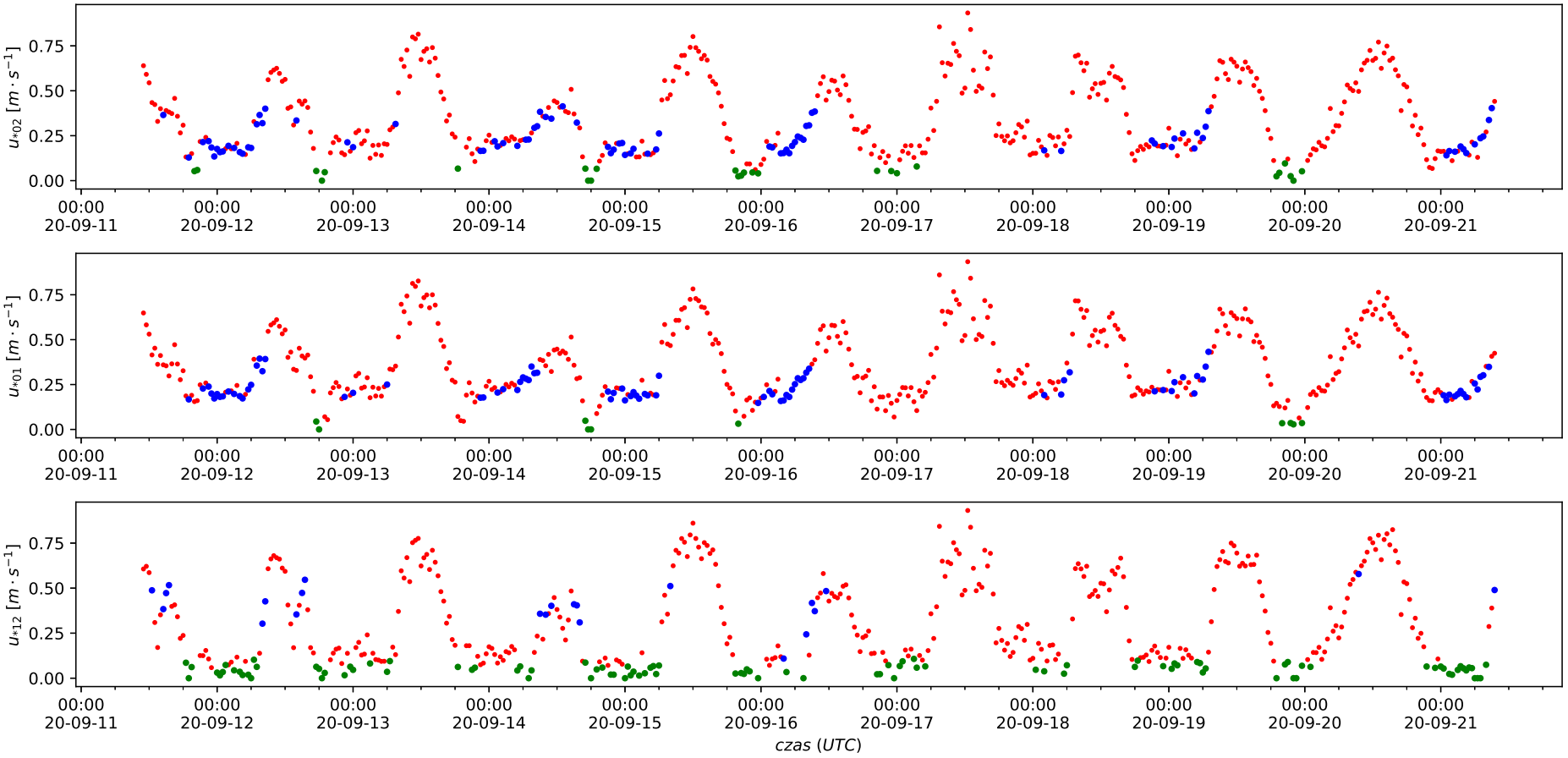
Rysunek 48: Turbulencyjne strumienie ciepła jawnego w dniach 11 września - 21 września 2020 roku wyznaczone na podstawie pomiarów między sensorami na poziomach 0 i 2 (H_{02}), 0 i 1 (H_{01}), 1 i 2 (H_{12}). Punkty niebieskie i zielone oznaczają wartości dla których parametr stabilności znajduje się poza zakresem stosowalności funkcji podobieństwa (niebieskie: $\zeta < -2$, zielone: $\zeta > 1$).



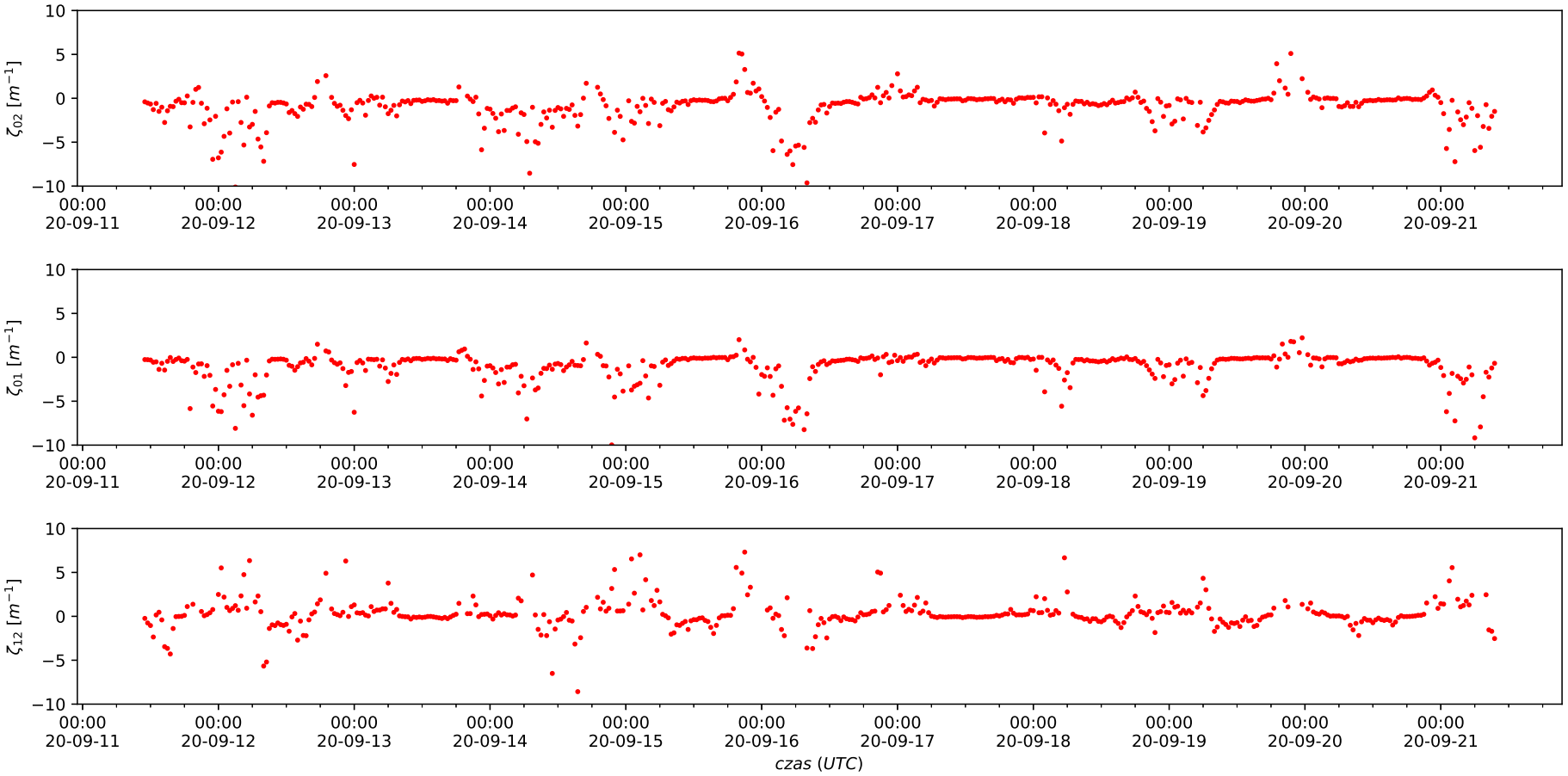
Rysunek 49: Turbulencyjne strumienie ciepła utajonego w dniach 11 września - 21 września 2020 roku wyznaczone na podstawie pomiarów między sensorami na poziomach 0 i 2 (E_{02}), 0 i 1 (E_{01}), 1 i 2 (E_{12}). Punkty niebieskie i zielone oznaczają wartości dla których parametr stabilności znajduje się poza zakresem stosowalności funkcji podobieństwa (niebieskie: $\zeta < -2$, zielone: $\zeta > 1$).



Rysunek 50: Turbulencyjne strumienie pędu w dniach 11 września - 21 września 2020 roku wyznaczone na podstawie pomiarów między sensorami na poziomach 0 i 2 (τ_{02}), 0 i 1 (τ_{01}), 1 i 2 (τ_{12}). Punkty niebieskie i zielone oznaczają wartości dla których parametr stabilności znajduje się poza zakresem stosowalności funkcji podobieństwa (niebieskie: $\zeta < -2$, zielone: $\zeta > 1$).



Rysunek 51: Prędkość tarciowa w dniach 11 września - 21 września 2020 roku wyznaczona na podstawie pomiarów między sensorami na poziomach 0 i 2 (u_{*02}), 0 i 1 (u_{*01}), 1 i 2 (u_{*12}). Punkty niebieskie i zielone oznaczają wartości dla których parametr stabilności znajduje się poza zakresem stosowalności funkcji podobieństwa (niebieskie: $\zeta < -2$, zielone: $\zeta > 1$).



Rysunek 52: Parametr stabilności ζ w dniach 11 września - 21 września 2020 roku wyznaczony na podstawie pomiarów między sensorami na poziomach 0 i 2 (ζ_{02}), 0 i 1 (ζ_{01}), 1 i 2 (ζ_{12}). Jako wysokość odniesienia przyjęto poziom wyższego sensora mierzony od wysokości płaszczyzny przesunięcia profilu wiatru d.

W dniach 19 września - 21 września 2020 roku, w celu przetestowania poprawności uzyskanych wyników, wykonano dodatkowo pomiary turbulencyjnych strumieni pędu i ciepła jawnego metodą kowariancji wirów, przy użyciu anemometru akustycznego producenta Gill Instruments WindMaster Pro [32] [3]. Niestety, z powodu nieprawidłowego działania anemometru, nie udało uzyskać się danych w całym zakresie czasowym pomiaru. Wartości otrzymanych w ten sposób strumieni i prędkości tarciowej przedstawia rysunek 53. Korzystając z zależności 4.81 - 4.84, opisanych w rozdziale *Kalibracja układu*, oszacowano parametry dopasowania pomiarów wykonanych metodą gradientową (między sensorami na poziomach 0 i 2 oraz 0 i 1) w stosunku do pomiarów wykonanych metodą kowariancji wirów. Pominięto wyznaczanie dopasowania pomiarów między poziomami 1 i 2, ponieważ je odrzucono ze względu na ich nierealistycznie duże wartości. Dopasowane proste przedstawiają rysunki 54 i 55. Dokładne parametry dopasowania przedstawiają tabele 10 i 11.

Tabela 10: Parametry dopasowania pomiarów metodą kowariancji wirów i metodą gradientową między sensorami na poziomach 0 i 2.

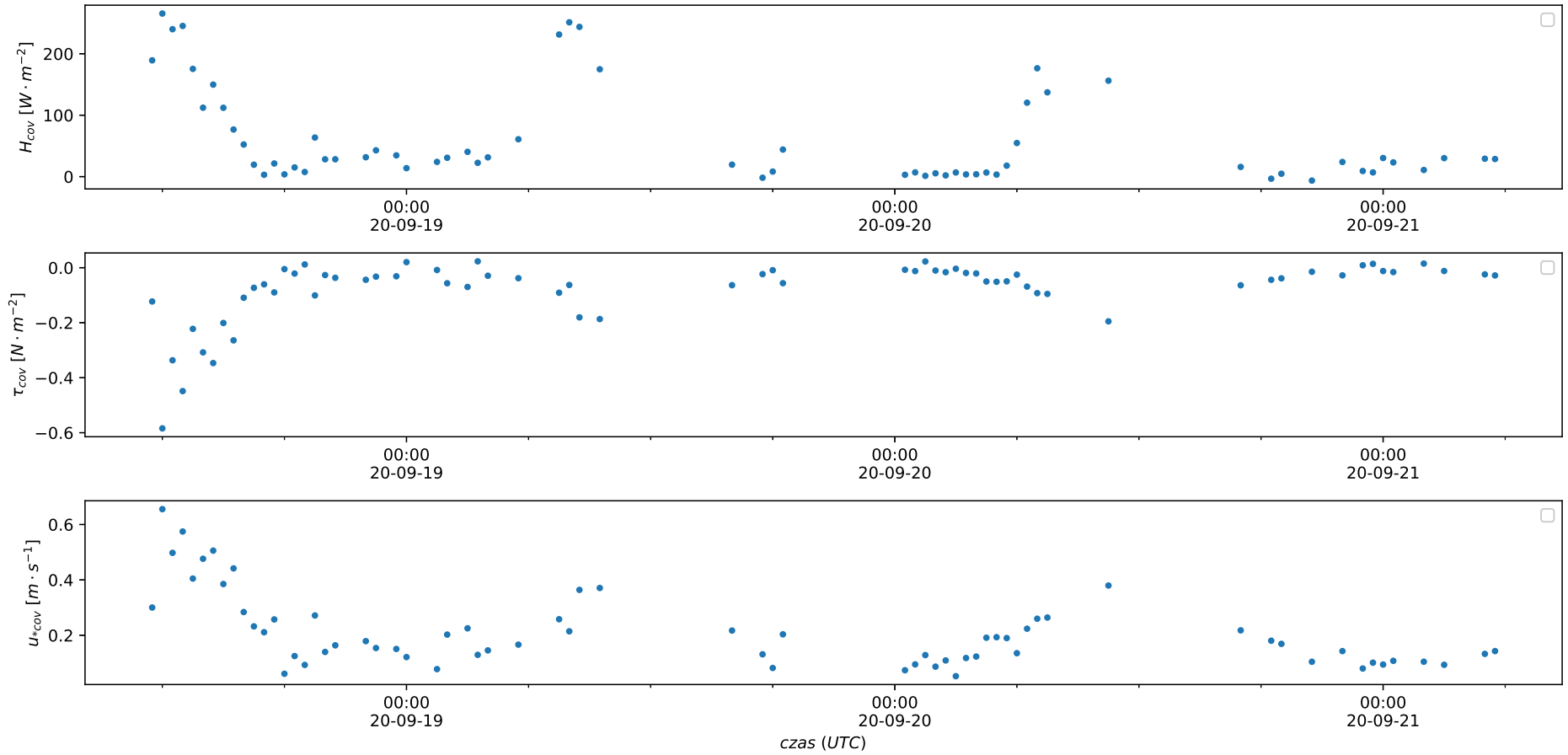
wielkość	współczynnik a	współczynnik b	współczynnik r	σ_a	σ_b
H	0.3947	13.5708	0.9370	0.0184	4.0252
τ	-0.5760	0.0037	-0.7368	0.0661	0.0135
u_*	0.6506	0.0149	0.8056	0.0598	0.0202

Tabela 11: Parametry dopasowania pomiarów metodą kowariancji wirów i metodą gradientową między sensorami na poziomach 1 i 2.

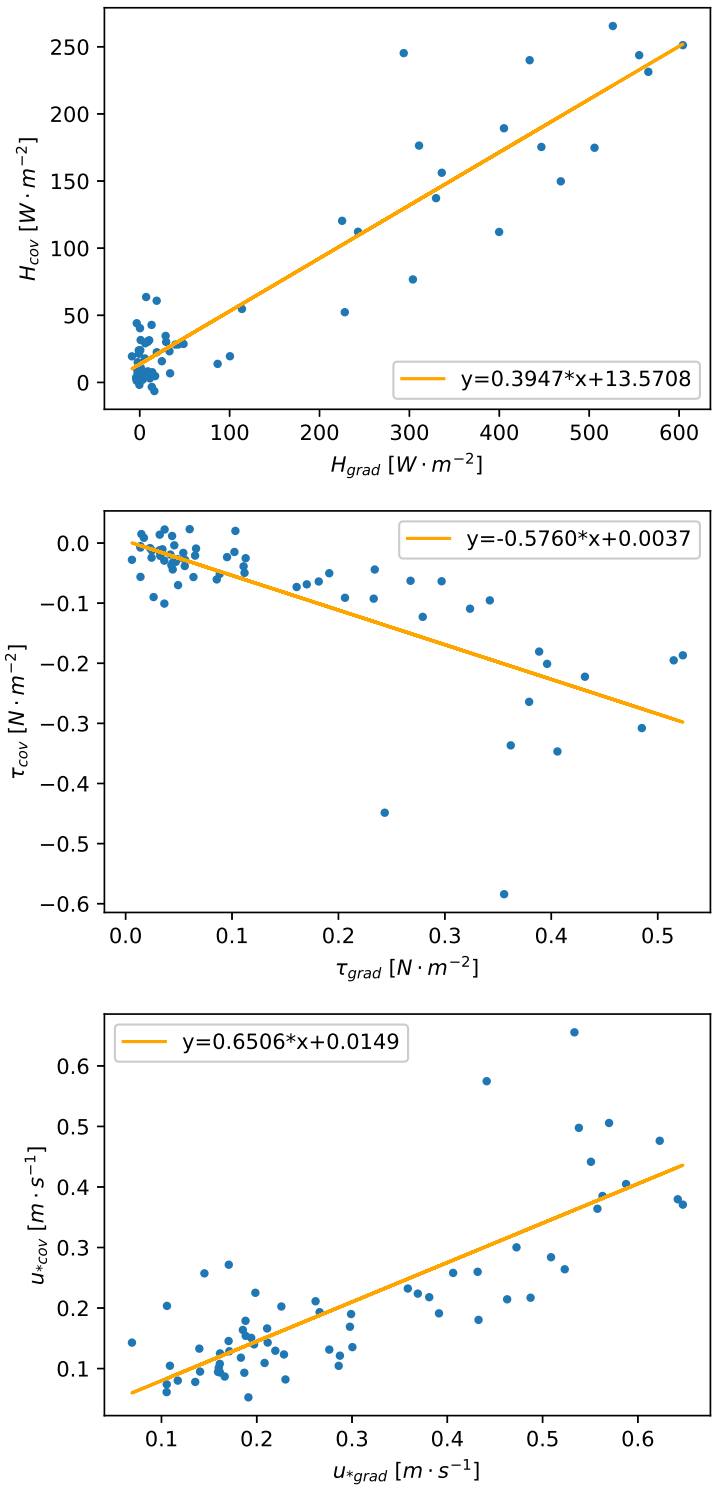
sensor	współczynnik a	współczynnik b	współczynnik r	σ_a	σ_b
H	0.3601	6.5664	0.9178	0.0192	4.6756
τ	-0.5994	0.0124	-0.7418	0.0667	0.0136
u_*	0.6969	-0.0109	0.8077	0.0626	0.0216

Ujemne współczynniki a w porównaniu parametrów τ wynikają z odwrotnego znaku strumienia turbulencyjnego pędu przyjętego w niniejszej pracy w stosunku do wyznaczonego w pomiarze przez anemometr.

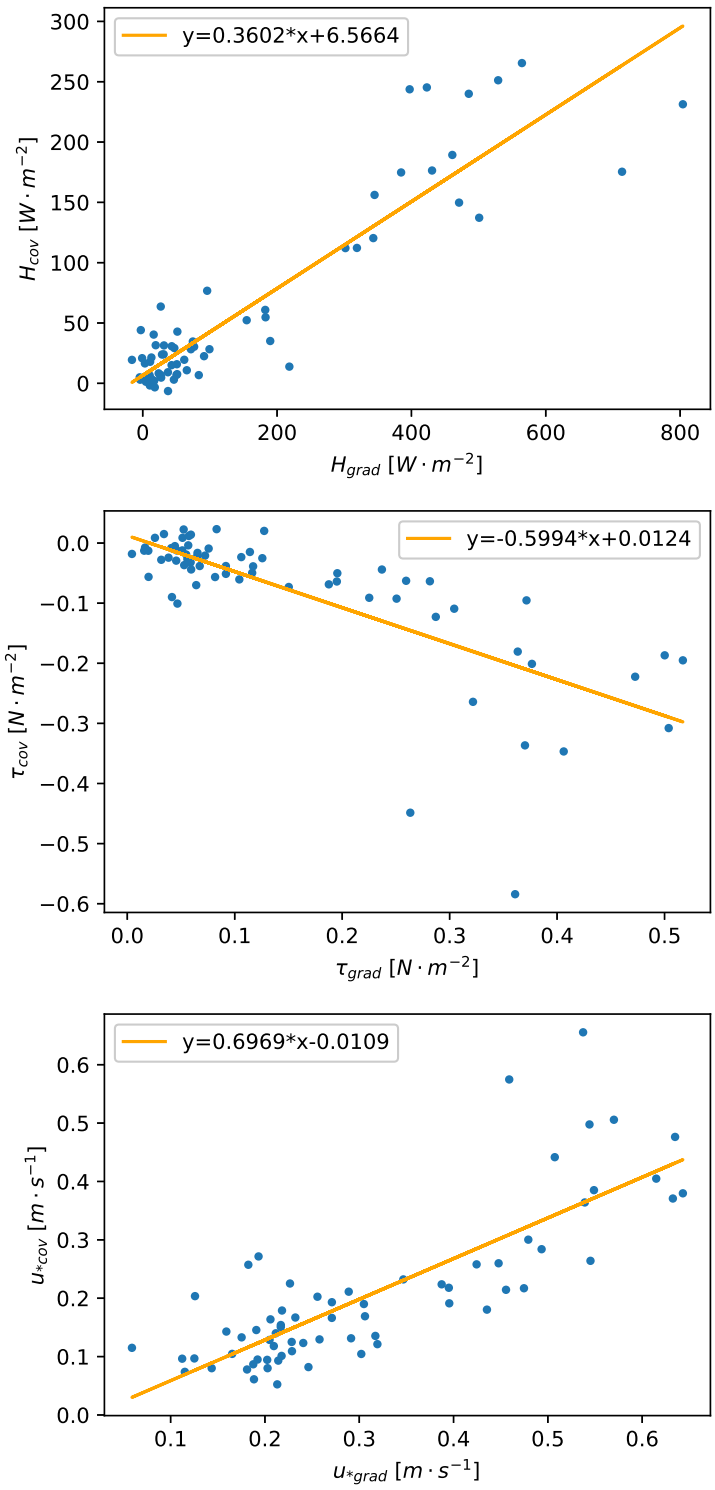
[3] Wyniki pomiarów dostarczył dr hab. inż. Mirosław Zimnoch, prof. AGH.



Rysunek 53: Wartości strumieni turbulencyjnych ciepła jawnego, pędu i prędkości tarciowej w dniach 18 września - 21 września zmierzonych metodą kowariancji wirów.



Rysunek 54: Porównanie pomiarów turbulencyjnych strumieni ciepła jawnego, pędu i prędkości tarciowej wykonanych metodą kowariancji wirów i metodą gradientową między sensorami na poziomach 0 i 2.



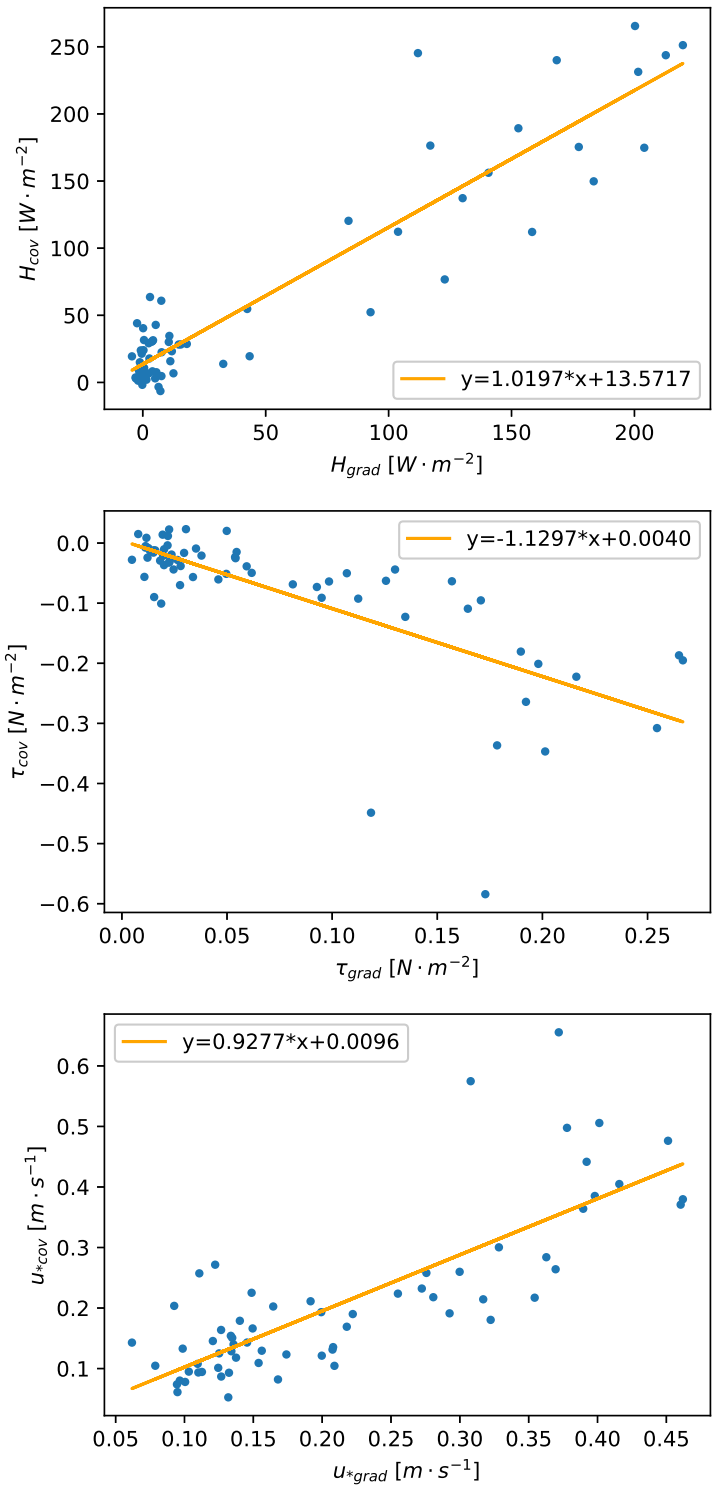
Rysunek 55: Porównanie pomiarów turbulencyjnych strumieni ciepła jawnego, pędu i prędkości tarciowej wykonanych metodą kowariancji wirów i metodą gradientową między sensorami na poziomach 1 i 2.

Słabe dopasowania prędkości tarciowej i turbulencyjnych strumieni pędu, które pomiarowo zależą wyłącznie od współczynników szorstkości terenu, przesunięcia płaszczyzny zerowej i prędkości wiatru (oraz poprawek ψ), którego profil na obszarze prowadzonych pomiarów wspomnianymi wyżej metodami powinien być w przybliżeniu taki sam, pozwalają sformułować hipotezę, wedle której wyznaczone w pracy [19] parametry otoczenia budynku wydziału w postaci przesunięcia płaszczyzny zerowej profilu wiatru $d = 18.56m$ i współczynnika szorstkości terenu $z_0 = 1.4m$ są już nie aktualne. Może to być spowodowane tym, że od momentu ich wyznaczenia doszło do istotnych zmian urbanistycznych okolicy budynku wydziału Fizyki i Informatyki Stosowanej. Tuż przy zachodniej ścianie budynku powstaje bowiem nowy budynek o z zbliżonej wysokości. Na tej podstawie postanowiono metodą *brute force* wyznaczyć nowe hipotetyczne wartości współczynników d i z_0. W tym celu iterowano po wszystkich możliwych kombinacjach wartości tych współczynników z krokiem $0.05m$, obliczano dla nich, na podstawie metody gradientowej, wartości strumieni H, τ i prędkości tarciowej u_* między sensorami 0 i 2, a następnie szukano najlepszego ich dopasowania do pomiarów metodą kowariancji wirów, tak aby współczynniki nachylenia prostych regresji liniowej były jak najbliższe wartości ± 1 w granicach błędu dopasowania. Najlepsze dopasowanie uzyskano dla następujących wartości parametrów: $d = 20.80m$ i $z_0 = 0.55m$. Parametry dopasowanych prostych przedstawia tabela 12, a dopasowane proste rysunek 56. Nowo obliczone na tej podstawie strumienie turbulencyjne ciepła jawnego, utajonego i pędu przedstawia rysunek 57.

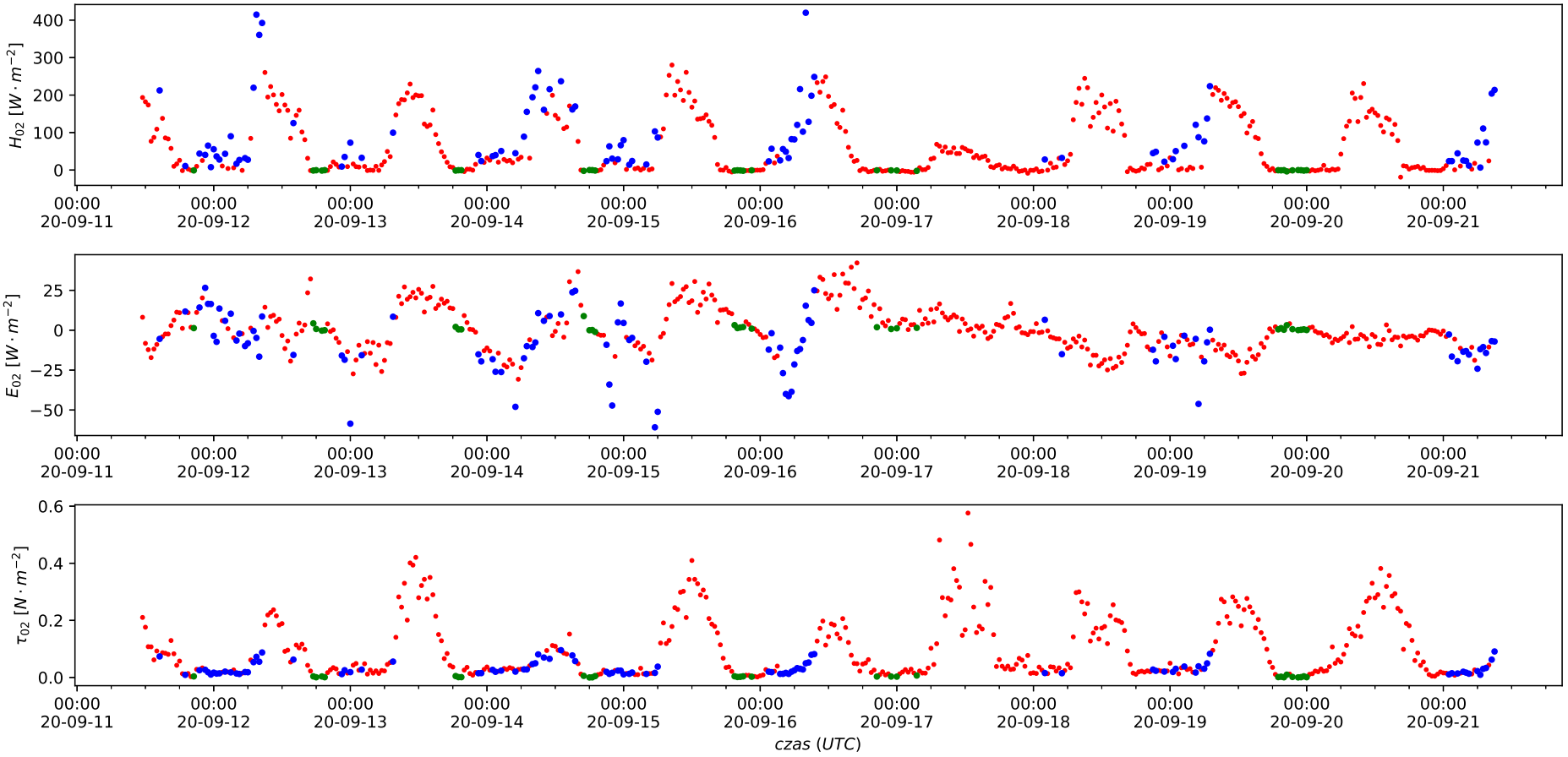
Tabela 12: Parametry najlepszego dopasowania pomiarów metodą kowariancji wirów i metodą gradientową między sensorami 0 i 2.

Wielkość	współczynnik a	współczynnik b	współczynnik r	σ_a	σ_b
H	1.0197	13.5717	0.9306	0.0501	4.2253
τ	-1.1297	0.0040	-0.7248	0.1342	0.0140
u_*	0.9277	0.0095	0.7998	0.0870	0.0211

Wyznaczone wartości parametrów d i z_0 powinny być jednak traktowane jako wartości hipotetyczne, gdyż liczba punktów pomiarowych z metody kowariancji wirów jest stosunkowo mała dla tej analizy.



Rysunek 56: Porównanie pomiarów turbulencyjnych strumieni ciepła jawnego, pędu i prędkości tarciowej wykonanych metodą kowariancji wirów i metodą gradientową między sensorami na poziomach 0 i 2 ze zmienionym współczynnikami d i z_0.



Rysunek 57: Strumienie turbulencyjne ciepła jawnego, utajonego i pędu w dniach 11 września - 21 września 2020 roku wyznaczone na podstawie pomiarów między sensorami na poziomach 0 i 2 ze zmienionym współczynnikami d i z_0. Punkty niebieskie i zielone oznaczają wartości dla których parametr stabilności znajduje się poza zakresem stosowalności funkcji podobieństwa (niebieskie: $\zeta < -2$, zielone: $\zeta > 1$).

Wartości turbulencyjnych strumieni ciepła jawnego, po zmodyfikowaniu wartości parametrów d i z_0, są zgodne z poziomami typowo obserwowanymi wartościami na terenie Polski, które osiągają swoje maksimum w ciągu dnia na poziome $300W \cdot m^{-2}$ w miesiącach letnich i nieco mniejsze w miesiącach jesiennych, a w nocy oscylują wokół zera [28] [18]. Niestety, z powodu braku możliwości sprzętowych, nie przetestowano poprawności pomiarów turbulencyjnych strumieni ciepła utajonego.

6 Podsumowanie

Celem niniejszej pracy było zaprojektowanie, zmontowanie i testowanie systemu do pomiaru metodą gradientową turbulencyjnych strumieni ciepła w planetarnej warstwie granicznej. System pomiarowy zmontowany został w oparciu o platformę Arduino i sensory BME280. Spełniono wszystkie wymagania funkcjonalne i pozafunkcjonalne przedstawione w podrozdziale *Specyfikacja wymagań systemowych*. Wszystkie potencjalne trudności i problemy, przewidywane na etapie projektowania, zostały pomyślnie rozwiązane. Pomiary temperatury, wilgotności i ciśnienia skalibrowane zostały względem stacji pogodowej Davis Instruments Vantage Pro. Po tej operacji system został umieszczony na maszcie na dachu budynku Wydziału Fizyki i Informatyki Stosowanej.

Obliczone metodą bulk transfer wartości strumieni na podstawie pomiarów temperatur i wilgotności wykonanych przez system, oraz pomiarów prędkości wiatru wykonanych przez wydziałową stację meteorologiczną, wykazują liniową korelację względem pomiarów referencyjnych wykonanych metodą kowariancji wirów. Konieczne jest jednak dalsze testowanie poprawności uzyskiwanych wyników na większej ilości danych, oraz przy wyznaczeniu nowych wartości parametrów przesunięcia płaszczyzny zerowej profilu wiatru i szorstkości terenu. Bardzo pomocne mogłoby się okazać również rozszerzenie pomiarów prędkości wiatru o pomiar na kolejnej wysokości. Ze względu na pojawiające się w literaturze kontrowersje dotyczące stosowania logarytmicznych profili wiatru na terenach zurbanizowanych, a przez to stosowania na nich metod gradientowych [18], należałoby również przeprowadzić testy wpływu stosowania innych modeli profilu wiatru na uzyskiwane wyniki. Należy również w przyszłości dokonać sprawdzenia poprawności wyników turbulencyjnych strumieni ciepła utajonego, które ze względu na brak możliwości technicznych nie zostały sprawdzone w niniejszej pracy.

Literatura

[1] Biblioteka do obsługi sensora BME280 producenta Adafruit:
https://github.com/adafruit/Adafruit_BME280_Library [Dostęp: 1 października 2020]

[2] Biblioteka do obsługi układu multipleksera producenta DFRobot Gravity:
https://github.com/DFRobot/DFRobot_I2C_Multiplexer [Dostęp: 1 października 2020]

[3] Biblioteka do obsługi zegara watchodg:
https://github.com/adafruit/Adafruit_SleepyDog [Dostęp: 1 października 2020]

[4] Dokumentacja biblioteki NumPy:
https://numpy.org/doc/ [Dostęp: 1 października 2020]

[5] Dokumentacja biblioteki Pandas:
https://pandas.pydata.org/docs/ [Dostęp: 1 października 2020]

[6] Dokumentacja języka Arduino:
https://www.arduino.cc/reference/en/ [Dostęp: 1 października 2020]

[7] Dokumentacja języka Python:
https://docs.python.org/3/ [Dostęp: 1 października 2020]

[8] Dokumentacja oprogramowanie EddyPro:
https://www.licor.com/documents/1ium2zmwm6hl36yz9bu4 [Dostęp: 1 października 2020]

[9] Dokumentacja sensora BME280:
https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf [Dostęp: 1 października 2020]

[10] Dokumentacja układu BSS138:
https://www.onsemi.com/pub/Collateral/BSS138-D.PDF [Dostęp: 1 października 2020]

[11] Dokumentacja układu DS1307:
https://datasheets.maximintegrated.com/en/ds/DS1307.pdf [Dostęp: 1 października 2020]

[12] Dokumentacja układu LM2596:
https://www.onsemi.com/pub/Collateral/LM2596-D.PDF [Dostęp: 1 października 2020]

[13] Dokumentacja układu NDC7002N:
https://www.onsemi.com/pub/Collateral/NDC7002N-D.pdf [Dostęp: 1 października 2020]

[14] Dokumentacja układu P82B715:
https://www.nxp.com/docs/en/data-sheet/P82B715.pdf [Dostęp: 1 października 2020]

[15] Dokumentacja układu RT9193:
https://www.richtek.com/assets/product_file/RT9193/DS9193-16.pdf [Dostęp: 1 października 2020]

[16] Dokumentacja układu TCA9548A:
https://www.ti.com/lit/ds/symlink/tca9548a.pdf [Dostęp: 1 października 2020]

[17] NXP Semiconductors: *I2C-bus specification and user manual*
https://www.nxp.com/docs/en/user-guide/UM10204.pdf [Dostęp: 1 października 2020]

[18] Fortuniak K.: *Radiacyjne i turbulencyjne składniki bilansu cieplnego terenów zurbanizowanych na przykładzie Łodzi.* Łódź: Wydawnictwo Uniwersytetu Łódzkiego, 2010

[19] Jasek-Kamińska A.: *Badanie zmienności strumienia i składu izotopowego biogenicznych emisji dwutlenku węgla do atmosfery na terenie aglomeracji krakowskiej.* Praca doktorska, Kraków: WFiIS AGH, 2017

[20] Karta katalogowa BiTsensor PE-PVC Blue 2x2x22AWG:
https://bitner.com.pl/data/t4txt_photos/113256240519_16_BiTsensor_PE-PVC_Blue.pdf [Dostęp: 1 października 2020]

[21] Karta katalogowa Helukabel SiHF 2x0.5:
http://centrala.helukabel.pl/proalpha/images/catalog/pl/22989.pdf [Dostęp: 1 października 2020]

[22] Karta katalogowa TFA 98.1114.02:
https://clientmedia.trade-server.net/1768_tfadost/media/5/50/5550.pdf [Dostęp: 1 października 2020]

[23] Launiainen J., Vihma T.: *Derivation of turbulent surface fluxes — An iterative flux-profile method allowing arbitrary observing heights.* Environmental Software 5(3), 113-124. 1990 doi:10.1016/0266-9838(90)90021-W

[24] Mielczarek W.: *Szeregowe interfejsy cyfrowe.* Gliwice: Helion, 1993

[25] Min-Seong K., Byung-Hyuk K.: *Estimation of Sensible Heat Flux and Atmospheric Boundary Layer Height Using an Unmanned Aerial Vehicle* Atmosphere 10(7):363 2019 doi:10.3390/atmos10070363

[26] Min-Seong K., Byung-Hyuk K., Dong-Hwan K.: *Estimation of Atmospheric Turbulent Fluxes by the Bulk Transfer Method over Various Surface.* Journal of Environmental Science International. 23. 1199-1211. 2014 doi:10.5322/JESI.2014.23.6.1199.

[27] Moene A., Dam J.: *Transport in the Atmosphere-Vegetation-Soil Continuum.* Cambridge: Cambridge University Press., 2014, doi:10.1017/CBO9781139043137

[28] Paszyński J., Miara K., Skoczek J.: *Wymiana energii między atmosferą a podłożem jako podstawa kartowania topoklimatycznego.* Warszawa: PAN IGiPZ, 1999

[29] Strona internetowa producenta Arduino MKRZero: https://www.arduino.cc/en/Guide/ArduinoMKRZero [Dostęp: 1 października 2020]

[30] Strona internetowa producenta Davis Instruments Vantage Pro: https://www.davisinstruments.com/product/cabled-vantage-pro-with-standard-radiation-shield/ [Dostęp: 1 października 2020]

[31] Strona internetowa producenta DFRobot Gravity: I2C Multiplexer: https://wiki.dfrobot.com/Gravity__Digital_1-to-8_I2C_Multiplexer_SKU_DFR0576 [Dostęp: 1 października 2020]

[32] Strona internetowa producenta Gill Instruments WindMaster Pro: http://www.gillinstruments.com/products/anemometer/windmaster-pro.html [Dostęp: 1 listopada 2020]

[33] Strona internetowa producenta SparkFun BOB-12009: https://www.sparkfun.com/products/12009 [Dostęp: 1 października 2020]

[34] Strona internetowa producenta Spelsberg Abox 025-L: https://www.spelsberg.pl/puszki-laczeniowe/obszar-wewnetrzny-ip65-do-702/80290701/ [Dostęp: 1 października 2020]

[35] Strona internetowa producenta Waveshare BME280 Environmental Sensor: https://www.waveshare.com/wiki/BME280_Environmental_Sensor [Dostęp: 1 października 2020]

[36] Strona internetowa producenta WEiPU SP13:
https://www.weipuconnector.com/product/37 [Dostęp: 1 października 2020]

[37] Wild M., Folini D., Schär C., Loeb N., Dutton E. G., König-Langlo G.: *The global energy balance from a surface perspective.* Climate Dynamics 40, 3107–3134, 2013, doi:DOI 10.1007/s00382-012-1569-8.

[38] Wiśniewski S., Wiśniewski T.S.: *Wymiana ciepła.* Warszawa: WNT, 1997

[39] Zespół Fizyki Środowiska: *System prezentacji pogody:* http://meteo.ftj.agh.edu.pl/ [Dostęp: 1 października 2020]

A Przykładowy fragment pliku z danymi pomiarowymi

nazwa pliku: `20200913.CSV`

kolumny: `data, godzina (UTC), numer sensora, temperatura, temperatura po kalibracji, wilgotność, wilgotność po kalibracji, ciśnienie, ciśnienie po kalibracji`

```
20200913,00:00:00,0,15.46,16.31,88.76,83.01,996.43,991.43
20200913,00:00:00,1,15.37,16.15,89.51,84.25,995.39,990.36
20200913,00:00:00,2,15.78,16.34,90.87,83.65,994.94,989.28
20200913,00:01:00,0,15.56,16.39,87.40,81.74,996.42,991.42
20200913,00:01:00,1,15.36,16.14,89.53,84.27,995.42,990.40
20200913,00:01:00,2,15.76,16.32,90.81,83.60,994.88,989.22
20200913,00:02:00,0,15.64,16.46,87.82,82.13,996.42,991.42
20200913,00:02:00,1,15.33,16.12,89.55,84.28,995.47,990.44
20200913,00:02:00,2,15.75,16.31,90.85,83.64,994.93,989.27
20200913,00:03:00,0,15.63,16.45,88.05,82.34,996.44,991.44
20200913,00:03:00,1,15.33,16.12,89.64,84.37,995.44,990.41
20200913,00:03:00,2,15.89,16.42,91.27,84.00,994.92,989.26
20200913,00:04:00,0,15.59,16.42,87.74,82.05,996.39,991.39
20200913,00:04:00,1,15.32,16.11,89.62,84.35,995.47,990.44
20200913,00:04:00,2,15.92,16.45,90.72,83.52,994.91,989.25
20200913,00:05:00,0,15.53,16.37,88.21,82.49,996.40,991.41
20200913,00:05:00,1,15.31,16.10,89.65,84.37,995.46,990.44
20200913,00:05:00,2,15.88,16.42,90.55,83.37,994.89,989.23
20200913,00:06:00,0,15.45,16.31,88.63,82.88,996.49,991.49
20200913,00:06:00,1,15.36,16.14,89.77,84.48,995.41,990.38
20200913,00:06:00,2,16.15,16.63,90.76,83.56,994.94,989.27
20200913,00:07:00,0,15.41,16.27,87.94,82.24,996.43,991.43
20200913,00:07:00,1,15.36,16.14,89.62,84.35,995.43,990.41
20200913,00:07:00,2,16.07,16.57,90.17,83.04,994.95,989.29
20200913,00:08:00,0,15.41,16.27,87.63,81.95,996.36,991.37
20200913,00:08:00,1,15.34,16.13,89.57,84.30,995.47,990.45
20200913,00:08:00,2,16.04,16.54,90.04,82.92,994.93,989.27
```

B Kod źródłowy sterownika systemu

```
#include <Wire.h>
#include <SD.h>
#include <SPI.h>
#include <RTClib.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Adafruit_SleepyDog.h>

int countDown = Watchdog.enable(30000); // 30 sec watchdog

RTC_DS1307 rtc;
Adafruit_BME280 bme0;
Adafruit_BME280 bme1;
Adafruit_BME280 bme2;
Adafruit_BME280 BME[3] = {
  bme0,
  bme1,
  bme2
};

bool sensorCheck[3];

uint8_t I2CMultiplexer = 0x70;
uint8_t RTCPort = 7;
uint8_t sensor0 = 0;
uint8_t sensor1 = 1;
uint8_t sensor2 = 2;

uint8_t sensorMin = sensor0;
uint8_t sensorMax = sensor2;

uint8_t LEDPin = 6;

const int chipSelect = SS1;

const int resetCounter = 360;
int counter = 0;

float sensorTemperatureCalibration[3][2] = {
  {0.81095174, 3.77583038},
  {0.80134696, 3.83381344},
  {0.7903676, 3.8657021}
};

float sensorHumidityCalibration[3][2] = {
  {0.93557591, -0.03525688},
  {0.90668954, 3.09000939},
  {0.87604735, 4.04745326}
};

float sensorPressureCalibration[3][2] = {
  {0.98785025, 7.10928356},
  {0.96861036, 26.22091315},
  {0.96370089, 30.45338835}
};

void selectPort(uint8_t port) {
  if (port > 7)
    return;

  Wire.beginTransmission(I2CMultiplexer);
  Wire.write(1 << port);
  Wire.endTransmission();
}

String getSensorValues(uint8_t port) {
```

```
  if (port > sensorMax || port < sensorMin)
    return "␣";

  selectPort(port);

  BME[port].takeForcedMeasurement();

  float temperature = BME[port].readTemperature();
  float humidity = BME[port].readHumidity();
  float pressure = BME[port].readPressure() / 100.0F;

  if(isnan(temperature) || isnan(humidity) || isnan(pressure))
    return "nan,nan,nan,nan,nan,nan";

  float temperatureCalibrated =
    temperature * sensorTemperatureCalibration[port][0] +
    sensorTemperatureCalibration[port][1];
  float humidityCalibrated =
    humidity * sensorHumidityCalibration[port][0] +
    sensorHumidityCalibration[port][1];
  float pressureCalibrated =
    pressure * sensorPressureCalibration[port][0] +
    sensorPressureCalibration[port][1];

  String dataString =
    String(temperature) + "," + String(temperatureCalibrated) + "," +
    String(humidity) + "," + String(humidityCalibrated) + "," +
    String(pressure) + "," + String(pressureCalibrated);

  return dataString;
}

String getTime() {
  selectPort(RTCPort);

  DateTime now = rtc.now();

  String timeString;

  uint8_t timeHour = now.hour();
  if (timeHour < 10)
    timeString += "0";
  timeString += String(timeHour) + ":";

  uint8_t timeMinute = now.minute();
  if (timeMinute < 10)
    timeString += "0";
  timeString += String(timeMinute) + ":";

  uint8_t timeSecond = now.second();
  if (timeSecond < 10)
    timeString += "0";
  timeString += String(timeSecond);

  return timeString;
}

uint8_t getHour(){
  selectPort(RTCPort);
  DateTime now = rtc.now();

  return now.hour();
}

uint8_t getMinute(){
  selectPort(RTCPort);
  DateTime now = rtc.now();
```

```
  return now.minute();
}

uint8_t getSecond(){
  selectPort(RTCPort);
  DateTime now = rtc.now();

  return now.second();
}

String getDate() {
  selectPort(RTCPort);
  DateTime now = rtc.now();
  String dateString = String(now.year());

  uint8_t dateMonth = now.month();
  if (dateMonth < 10)
    dateString += "0";
  dateString += String(dateMonth);

  uint8_t dateDay = now.day();
  if (dateDay < 10)
    dateString += "0";
  dateString += String(dateDay);

  return dateString;
}

void writeData(String dataString, String filename) {
  File dataFile = SD.open(filename, FILE_WRITE);
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
    Serial.println(dataString);
  } else {
    Serial.println("Data write error.");
    flashLED(5, 300);
  }
}

void flashLED(int number, int len) {
  for (int i = 0; i < number; i++) {
    digitalWrite(LEDPin, LOW);
    delay(len);
    digitalWrite(LEDPin, HIGH);
    delay(len);
    digitalWrite(LEDPin, LOW);
  }
}

void initializeSDCard() {
  while (!SD.begin(chipSelect)) {
    Serial.println("SD card allocation failed");
    flashLED(2, 300);
    delay(2000);
  }
  Serial.println("SD card initialized.");
  digitalWrite(LEDPin, LOW);

}

void initializeI2CMultiplexer() {
  Wire.begin();
  Serial.println("I2CMultiplexer initialized");
}

void initializeRTC() {
  selectPort(RTCPort);
```

```
  while (!rtc.begin() || !rtc.isrunning()) {
    Serial.println("RTC initialization failed");
    flashLED(3, 300);
    delay(2000);
  }
  Serial.println("RTC initialized");
}

void initializeSensors() {
  for (uint8_t i = sensor0; i <= sensor2; i++) {
    selectPort(i);
    for (int j = 0; j < 5; j++) {
      sensorCheck[i] = BME[i].begin(0x77, &Wire);
      if (sensorCheck[i]) {

        BME[i].setSampling(
          Adafruit_BME280::MODE_FORCED,
          Adafruit_BME280::SAMPLING_X1, // temperature
          Adafruit_BME280::SAMPLING_X1, // pressure
          Adafruit_BME280::SAMPLING_X1, // humidity
          Adafruit_BME280::FILTER_OFF
        );
        continue;

      }
      else {
        String log_data =
          getDate() + " "  + getTime() + " Sensor " + String(i) +
          " initialization failed (" + String(j + 1) + ")";
        Serial.println(log_data);
        writeData(log_data, "info.LOG");
        flashLED(4, 300);
        delay(1000);
      }
    }
    digitalWrite(LEDPin, LOW);
  }
}

void initializeAll() {
  flashLED(5, 50);

  initializeI2CMultiplexer();
  initializeRTC();
  initializeSDCard();

  String log_data = getDate() + " "  + getTime() + " System turn-on/reset";
  Serial.println(log_data);
  writeData(log_data, "info.LOG");

  initializeSensors();

  counter = 0;

  flashLED(5, 50);
}

void setup() {
  Serial.begin(9600);

  pinMode(LEDPin, OUTPUT);
  initializeAll();
}

void loop() {

  digitalWrite(LEDPin, HIGH);
```

```
  String sensorData;
  String file;

  String currentTime = getTime();
  String currentDate = getDate();

  for (uint8_t i = sensorMin; i <= sensorMax; i++) {
    Watchdog.reset(); // Watchdog timer reset
    if (!sensorCheck[i]) {
      flashLED(4, 300);
      digitalWrite(LEDPin, HIGH);
      continue;
    }
    sensorData = getSensorValues(i);
    file = currentDate + ".CSV";
    writeData(
      currentDate + "," + currentTime + "," +
      String(i) + "," + sensorData, file
    );
  }

  counter++;
  if (counter == resetCounter) {
    flashLED(10, 10);
    String log_data =
      getDate() + "␣"  + getTime() + "␣Scheduled␣System␣reinitialization␣begin";
    Serial.println(log_data);
    writeData(log_data, "info.LOG");
    initializeAll();
  }

  delay(2000);
  digitalWrite(LEDPin, LOW);

  uint8_t dataReadMinute = getMinute();
  while(getMinute() == dataReadMinute){
    Watchdog.reset(); // Watchdog timer reset
    delay(1000);
    flashLED(1, 10);
  }
}
```

C Kod źródłowy skryptu kalibracyjnego

Nazwa skryptu: `calib.py`

Uruchomienie z linii poleceń:

`python calib.py -d DAVIS -c CALIB -t TIMEDELTA`

gdzie: `DAVIS` - ścieżka do raportu pogodowego stacji Davis w formacie xlsx, `CALIB` - ścieżka do folderu z pomiarami w postaci plików csv, `TIMEDELTA` - przesunięcie rejestrowanego czasu

przykład:

`python calib.py -d "./davis.xlsx" -c "./kalibracja" -t "19 hours"`

```
import argparse
import numpy as np
import pandas as pd

from pathlib import Path


parser = argparse.ArgumentParser(description='System calibration script')
parser.add_argument('-d', '--davis', type=str, help='Path to Davis xlsx report', required=True)
parser.add_argument('-c', '--calib', type=str, help='Path to folder with system CSV files', required=True)
parser.add_argument('-t', '--timedelta', type=str, help='Timedelta for time measurements', required=True)
args = parser.parse_args()

davis_path = Path(args.davis)
sensors_data_path = Path(args.calib).glob("*.CSV")

davis_data = pd.read_excel(davis_path, header=None, skiprows=[0, 1])
davis_data = davis_data[[0, 1, 2, 5, 15]]
davis_data.columns = ["Date_davis", "Time_davis", "Temp_davis", "Hum_davis", "Bar_davis"]
davis_data = davis_data[davis_data["Temp_davis"] != "---"]
davis_data = davis_data[davis_data["Hum_davis"] != "---"]
davis_data = davis_data[davis_data["Bar_davis"] != "---"]
davis_data[["Temp_davis", "Hum_davis", "Bar_davis"]] = davis_data[["Temp_davis",
                                                                   "Hum_davis",
                                                                   "Bar_davis"]].astype("float64")
davis_data["Timestamp_davis"] = pd.to_datetime(
    davis_data["Date_davis"].astype(str) + " " + davis_data["Time_davis"].astype(str)
)
davis_data["Timestamp_davis"] += pd.Timedelta(args.timedelta)

sensors_data = [pd.read_csv(file, header=None) for file in sensors_data_path]
sensors_data = pd.concat(sensors_data) if len(sensors_data) > 1 else sensors_data[0]
sensors_data.columns = ["Date", "Time", "Sensor", "Temp", "Hum", "Bar"]
sensors_data["Timestamp"] = pd.to_datetime(
    sensors_data["Date"].astype(str) + " " + sensors_data["Time"].astype(str)
)
sensors_data[["Temp", "Hum", "Bar"]] = sensors_data[["Temp", "Hum", "Bar"]].astype("float64")
sensors_data = sensors_data.sort_values(["Timestamp", "Sensor"])
sensors_data = [sensors_data[sensors_data["Sensor"] == i].dropna() for i in range(3)]
sensors_data = [
    pd.merge_asof(
        sensors_data[i], davis_data, left_on="Timestamp",
        right_on="Timestamp_davis", tolerance=pd.Timedelta("1 min")
    ).dropna()
    for i in range(3)
]
```

```
temp_calib = [
    [
        np.polyfit(sensors_data[i]["Temp"], sensors_data[i]["Temp_davis"], 1, cov=True),
        np.corrcoef(sensors_data[i]["Temp"], sensors_data[i]["Temp_davis"])
    ]
    for i in range(3)
]

hum_calib = [
    [
        np.polyfit(sensors_data[i]["Hum"], sensors_data[i]["Hum_davis"], 1, cov=True),
        np.corrcoef(sensors_data[i]["Hum"], sensors_data[i]["Hum_davis"])
    ]
    for i in range(3)
]

bar_calib = [
    [
        np.polyfit(sensors_data[i]["Bar"], sensors_data[i]["Bar_davis"], 1, cov=True),
        np.corrcoef(sensors_data[i]["Bar"], sensors_data[i]["Bar_davis"])
    ]
    for i in range(3)
]

print("Calib._type\t\tSensor\t\ta\t\tb\t\terr_a\t\terr_b\t\tr")
for i in range(3):
    print(f"Temperature\t\t{i}\t\t{temp_calib[i][0][0][0]:4.4f}\t\t"
          f"{temp_calib[i][0][0][1]:4.4f}\t\t"
          f"{np.sqrt(temp_calib[i][0][1][0][0]):4.4f}\t\t"
          f"{np.sqrt(temp_calib[i][0][1][1][1]):4.4f}\t\t"
          f"{temp_calib[i][1][0][1]:4.4f}")
    print(f"Humidity\t\t{i}\t\t{hum_calib[i][0][0][0]:4.4f}\t\t"
          f"{hum_calib[i][0][0][1]:4.4f}\t\t"
          f"{np.sqrt(hum_calib[i][0][1][0][0]):4.4f}\t\t"
          f"{np.sqrt(hum_calib[i][0][1][1][1]):4.4f}\t\t"
          f"{hum_calib[i][1][0][1]:4.4f}")
    print(f"Pressure\t\t{i}\t\t{bar_calib[i][0][0][0]:4.4f}\t\t"
          f"{bar_calib[i][0][0][1]:4.4f}\t\t"
          f"{np.sqrt(bar_calib[i][0][1][0][0]):4.4f}\t\t"
          f"{np.sqrt(bar_calib[i][0][1][1][1]):4.4f}\t\t"
          f"{bar_calib[i][1][0][1]:4.4f}")
```

D Kod źródłowy skryptu obliczeniowego do metody bulk transfer

Nazwa skryptu: `calc.py`

Uruchomienie z linii poleceń:

`python ./calc.py -D POMIAR -W WIATR -c CONFIG`

gdzie: `POMIAR` - ścieżka do folderu z pomiarami z systemu w postaci plików csv, `WIATR` - ścieżka do pliku csv z pomiarami prędkości wiatru z wydziałowej stacji meteorologicznej, `CONFIG` - ścieżka do pliku konfiguracyjnego

Struktura pliku konfiguracyjnego:

`d,18.56`

`z0,1.40`

`z,25.00`

`h0,21.98`

`h1,30.56`

`h2,39.70`

gdzie: `d` - przesunięcie płaszczyzny zerowej, `z0` - szorstkość terenu, `z` - wysokość pomiaru prędkości wiatru nad poziomem gruntu, `h0, h1, h1` - wysokości sensorów nad poziomem gruntu.

Wynik programu:

Pliki csv z przeliczonymi strumieniami turbulencyjnymi pędu, ciepła jawnego, ciepła utajonego, parametru stabilności, długości L i prędkości tarciowej między każdymi sensorami.

przykład uruchomienia:

`python ./calc.py -D "./pomiar" -W "./meteo.csv" -c "./config.cfg"`

```
import argparse
import numpy as np
import pandas as pd

from pathlib import Path

parser = argparse.ArgumentParser(description='Bulk method calculation script')
parser.add_argument('-D', '--data', type=str, help='Path to folder with system CSV files', required=True)
parser.add_argument('-W', '--wind', type=str,
                    help='Path to CSV file with wind speed values from WFiIS weather station', required=True)
parser.add_argument('-c', '--config', type=str, help='Path to config file', required=True)
args = parser.parse_args()
```

```
sensors_data_path = Path(args.data).glob("*.CSV")
wind_data_path = Path(args.wind)
cfg = pd.read_csv(Path(args.config), index_col=0, header=None).astype("float64")
print(cfg)
z = cfg[1]["z"]
d = cfg[1]["d"]
z0 = cfg[1]["z0"]
h2 = cfg[1]["h2"]
h1 = cfg[1]["h1"]
h0 = cfg[1]["h0"]
h = [h0, h1, h2]

k = 0.4
g = 9.81

c_p = 1005
rho = 1.25
l = 2257000

wind_data = pd.read_csv(wind_data_path)
wind_data = wind_data[["time", "sm"]]
wind_data["time"] = pd.to_datetime(wind_data["time"].astype(str)).dt.tz_localize(None)
wind_data["sm"] = wind_data["sm"].astype("float64")
wind_data = wind_data.sort_values("time")

sensors_data = [pd.read_csv(file, header=None) for file in sensors_data_path]
sensors_data = pd.concat(sensors_data) \
    if len(sensors_data) > 1 else sensors_data[0]
sensors_data.columns = [
    "Date", "Time", "Sensor", "Temp",
    "Temp_calib", "Hum", "Hum_calib", "Bar", "Bar_calib"
]
sensors_data["Timestamp"] = pd.to_datetime(
    sensors_data["Date"].astype(str) + "␣" + sensors_data["Time"].astype(str)
)

sensors_data[["Temp", "Temp_calib", "Hum", "Hum_calib", "Bar", "Bar_calib"]] = \
    sensors_data[["Temp", "Temp_calib", "Hum", "Hum_calib", "Bar", "Bar_calib"]].astype("float64")

sensors_data = sensors_data.sort_values(["Timestamp", "Sensor"]).drop(["Date", "Time"], axis=1)

sensors_data = [
    sensors_data[sensors_data["Sensor"] == i]
    for i in range(3)
]

sensors_data = [
    pd.merge_asof(
        sensors_data[i], wind_data, left_on="Timestamp",
        right_on="time", tolerance=pd.Timedelta("60sec")
    )
    for i in range(3)
]

# SPEC HUM CALC
for i in range(3):
    sensors_data[i]["Hum_specific"] = sensors_data[i]["Temp_calib"].copy()

    sensors_data[i]["Hum_specific"] = \
        np.exp((-6763.6/(sensors_data[i][sensors_data[i]["Temp_calib"] > 0]["Temp_calib"] + 273.15)) - 4.9283
               * np.log(sensors_data[i][sensors_data[i]["Temp_calib"] > 0]["Temp_calib"] + 273.15) + 54.23)

    sensors_data[i]["Hum_specific"] = \
        np.exp((-6141 / (sensors_data[i][sensors_data[i]["Temp_calib"] > 0]["Temp_calib"] + 273.15)) + 24.3)

    sensors_data[i]["Hum_specific"] = sensors_data[i]["Hum_specific"] * sensors_data[i]["Hum_calib"] / 100
```

```
    sensors_data[i]["Hum_specific"] = \
        0.622 * sensors_data[i]["Hum_specific"] / (sensors_data[i]["Bar_calib"] - 0.378
                                                   * sensors_data[i]["Hum_specific"])

sensors_data = [
    sensors_data[i].groupby(pd.Grouper(key="Timestamp", freq='30min', label="right"))
    for i in range(3)
]
sensors_data = [sensors_data[i].mean() for i in range(3)]

# ITER
for sensor in [[0, 2], [0, 1], [1, 2]]:
    H_vals = []
    E_vals = []
    Tau_vals = []
    zeta_vals = []
    L_vals = []
    u_star_vals = []
    for i in range(len(sensors_data[0])):
        L = 10e15
        iters = 0

        psi_m_z = 0
        psi_m_z0 = 0
        psi_h_t1 = 0
        psi_h_t0 = 0

        dT = sensors_data[sensor[1]]["Temp_calib"][i] - sensors_data[sensor[0]]["Temp_calib"][i]
        dq = sensors_data[sensor[1]]["Hum_specific"][i] - sensors_data[sensor[0]]["Hum_specific"][i]
        T_m = (sensors_data[sensor[0]]["Temp_calib"][i] +
               sensors_data[sensor[1]]["Temp_calib"][i]) / 2 + 273.15
        u = sensors_data[0]["sm"][i]

        while True:
            if L >= 0:
                psi_m_z = -5 * (z - d) / L
                psi_m_z0 = -5 * z0 / L

                psi_h_t1 = -5 * (h[sensor[1]] - d) / L
                psi_h_t0 = -5 * (h[sensor[0]] - d) / L

            else:
                x = (1 - 15 * (z - d) / L) ** (-1 / 4)
                psi_m_z = 2 * np.log((1 + x ** -1) / 2) + np.log((1 + x ** -2) / 2) - 2 * np.arctan(
                    x ** -1) + np.pi / 2
                x = (1 - 15 * z0 / L) ** (-1 / 4)
                psi_m_z0 = 2 * np.log((1 + x ** -1) / 2) + np.log((1 + x ** -2) / 2) - 2 * np.arctan(
                    x ** -1) + np.pi / 2

                x = (1 - 15 * (h[sensor[1]] - d) / L) ** (-1 / 2)
                psi_h_t1 = 2 * np.log((1 + x ** -1) / 2)

                x = (1 - 15 * (h[sensor[0]] - d) / L) ** (-1 / 2)
                psi_h_t0 = 2 * np.log((1 + x ** -1) / 2)
            L_old = L

            u_star = k * u / (np.log((z - d) / z0) - psi_m_z + psi_m_z0)
            t_star = k * dT / (np.log((h[sensor[1]] - d) / (h[sensor[0]] - d)) - psi_h_t1 + psi_h_t0)
            q_star = k * dq / (np.log((h[sensor[1]] - d) / (h[sensor[0]] - d)) - psi_h_t1 + psi_h_t0)

            L = u_star ** 2 * T_m / (k * g * t_star)

            if abs(L_old - L) < 10e-5:
                Tau = rho * u_star ** 2
                H = -rho * c_p * u_star * t_star
                E = -rho * l * u_star * q_star
                H_vals.append(H)
                E_vals.append(E)
```

```
                Tau_vals.append(Tau)
                L_vals.append(L)
                zeta_vals.append((h[sensor[1]] - d) / L)
                u_star_vals.append(u_star)
                break
            elif iters > 25:
                H_vals.append(np.nan)
                E_vals.append(np.nan)
                Tau_vals.append(np.nan)
                L_vals.append(np.nan)
                zeta_vals.append(np.nan)
                u_star_vals.append(np.nan)
                break
            iters += 1

    H_vals = pd.DataFrame(
        data={"H": H_vals, "E": E_vals, "Tau": Tau_vals, "(z1-d)/L": zeta_vals,
              "L": L_vals, "u_star": u_star_vals}, index=sensors_data[0].index)
    H_vals.to_csv(f"lvl_{sensor[0]}_{sensor[1]}.csv")
```