Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Individualaus darbo ataskaita

Individualaus „Įterptinių sistemų“ darbo ataskaita

Simonas Riauka

Studentas

Prof. Žilvinas Nakutis

Dėstytojas

Kaunas, 2023

Turinys

Įvadas ... **3**
1. Principinė schema ... **4**
1.1. Blokinė schemos diagrama ... 4
1.2. Komponentų pagrindimas ... 7
2. Programos schema ... **9**
2.1. Algoritmo aprašas ... 9
2.2. Matematinis pagrindimas ... 10
2.2.1. Įprastas šiaurės krypties skaičiavimo metodas ... 10
2.2.2. Atkompensuotos šiaurės krypties skaičiavimo metodas ... 11
2.3. Vartotojo kalibracijos aprašymas ... 13
2.4. Kompiuterinės programos galimybės ... 13
2.5. Komunikacijų su kompiuteriu aprašas ... 15
3. Testavimas ir rezultatai ... **17**
3.1. Testavimas ... 17
3.1.1. Fizinis veikimo testavimas ... 20
3.1.2. Kompiuterinės programos veikimo testavimas ... 23
3.2. Paklaidų vertinimas ... 25
3.2.1. Sensoriaus paklaidos ... 25
3.2.2. Apvalinimo paklaidos ... 27
3.2.3. Rezoliucijos paklaidos ... 27
Išvados ... **28**
Šaltiniai ... **29**
Priedai ... **30**
1 Mikrokontrolerio kodas ... 30
2 Kompiuterio programos kodas ... 36

Įvadas

Pagal užduoties variantą man priklausytų 11 užduotis, bet kadangi turiu STM maketo plokštę su integruotu e-kompaso jutikliu, pasinaudosiu šiais, jau turimais įrenginiais.

Užduoties tikslas: sukurti elektroninį kompaso atitikmenį vartotojui atvaizduojant virtualią kompaso rodyklę ir jos skirtumą nuo geografinės šiaurės. Šio projekto įgyvendinimui reikės pasinaudoti ir magnetometro ir pagreičio sensorių duomenimis, kurie bus skirti kompaso nehorizontaliam laikymui atkompensuoti ir teisingoms vertėms parodyti bet kokioje orientacijoje.

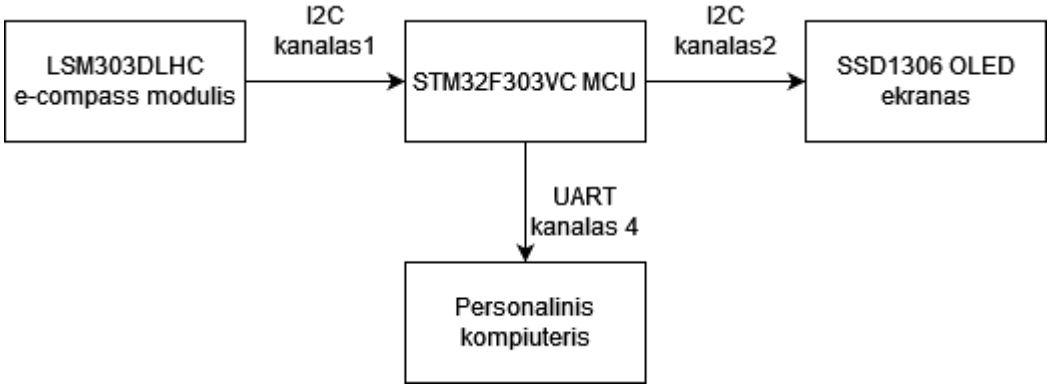
Lentelė 1. Sugalvotos užduoties varianto nurodymai

Matuojamas fizikinis dydis	Kanalų skaičius	Vaizduojami ir perduodami į kompiuterį parametrai	Signalo pralaidumo juosta	Dydžio diapazonas	Parodymų atnaujinimo indikatoriuje periodas, s	Parametrų diskretizavimo periodas, s
3D pagreičio ir 3D magnetometro sensoriai	1	Nuokrypio nuo šiaurės vidurkis	0-10Hz	0° - 360°	0.5	0.1

1. Principinė schema

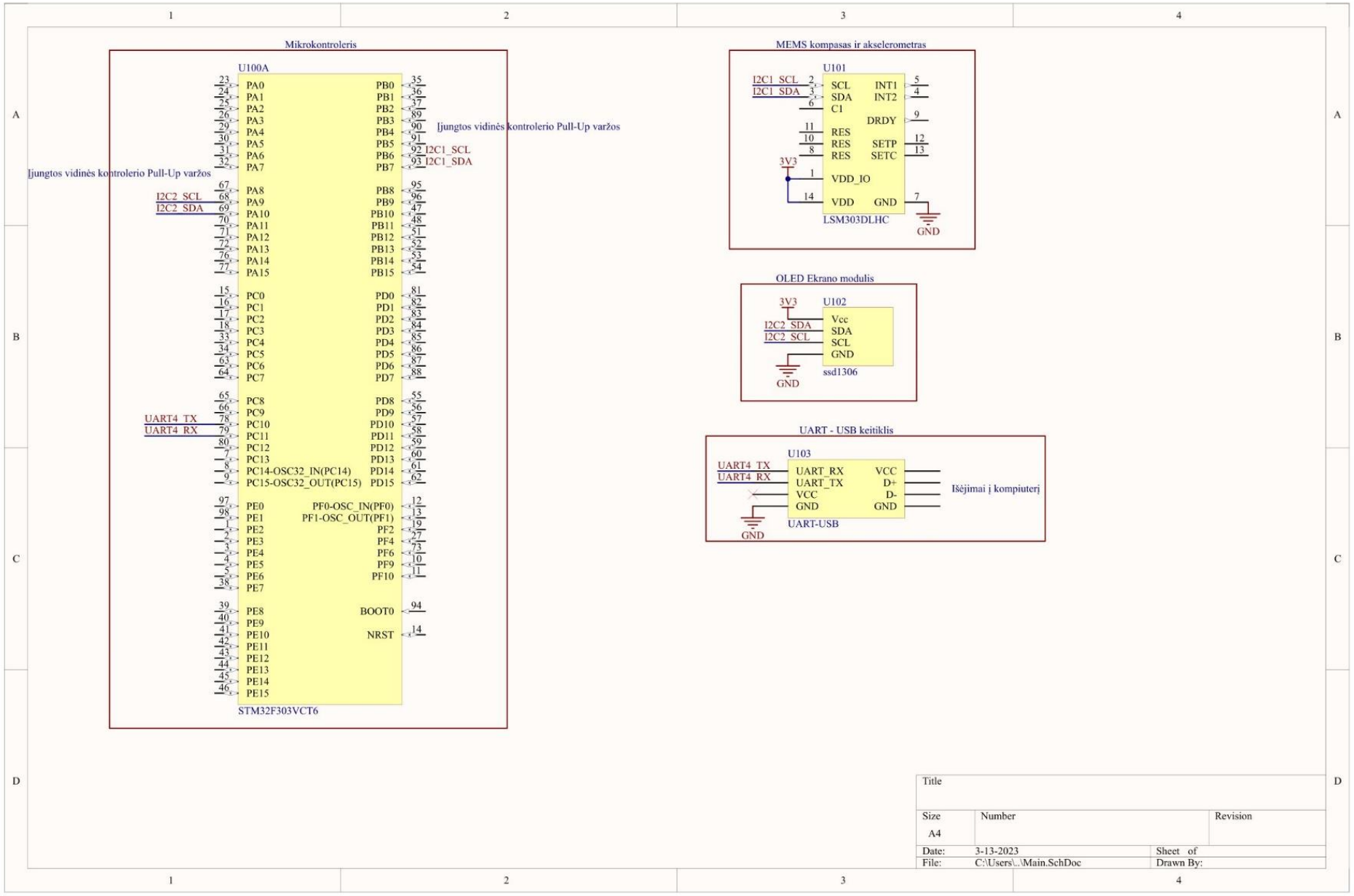
1.1. Blokinė schemos diagrama

Naudosime vieną modulį, kuriame yra integruoti magnetometro ir akselerometro sensoriai, jį nuskaitysime I2C sąsaja. OLED ekrano komunikacijoms bus panaudotas atskiras I2C kanalas dėl kitokio reikalaujamo sąsajos greičio.



1 pav. Įrenginio blokinė schema

Nubrėžta schema per Altium Designer programinę CAD įrangą:



2 pav. Įrenginio jungimo schema

E-kompaso modulis yra integruotas į maketą, todėl jo I2C kanalas nekeičiamas. OLED ekranas jungtas prie atskiro I2C kanalo, nes jam reikia greitojo 400kHz režimo, su kompiuteriu bendravimas per UART į USB keitiklį, UART4 kanalą lengvesniam fiziniam prijungimui, pasirinktas baud rate – 9600bit/s.

Lentelė 2. BOM

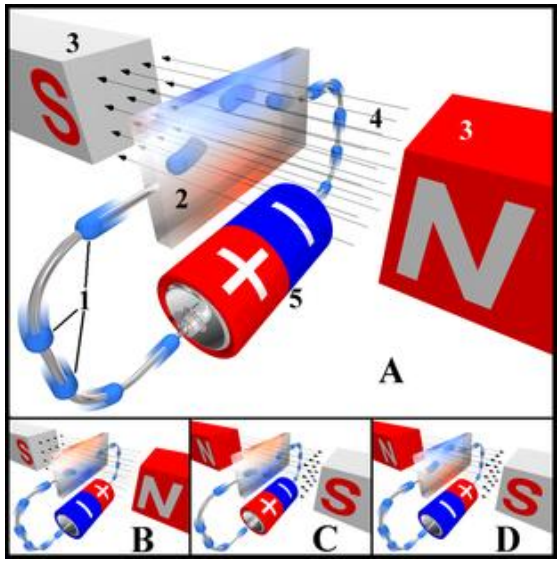
Pavadinimas	Komentaras	Kiekis	Nuoroda	Kaina, Eur
STM32F303VC DISCOVERY KIT	Mikrokontrolerio plokštė su integruotu LSM303DLHC e-kompaso sensoriumi	1	https://www.anodas.lt/stm32f3-discovery-stm32f3discovery	30.1
SSD1306	OLED ekranas	1	https://www.anodas.lt/oled-ekranas-melyna-grafika-0-96-quot-128x64px-i2c-melynas?search=SSD1306	9.50
PL2303	UART/USB konverteris	1	https://www.anodas.lt/keitiklis-usb-uart-pl2303-usb-kistukas-waveshare-4037?search=PL2303	7.50
Male to Female laidai	Laidai skirti sujungti mikrokontrolerio išvadus su OLED ekranu ir UART/USB konverteriu	7 (1 pakuotė)	https://www.lemona.lt/jungiamieji-laidai-su-kistukais-lizdais-15cm-10vnt-m-f-skirtingu-spalvu.html	2.20
USB-A į USB-B mini kabelis	Laidas skirtas tiekti elektrai į mikrokontrolerio plokštę	1	https://www.lemona.lt/kabelis-usb-a-kistukas-mini-usb-b-kistukas-1-0m-inakustik-premium.html	9.00
		Viso:		58.3

(Komponentai buvo pirkti seniau, su kitokia kaina ir ne visus išėjo rasti internetinėse parduotuvėse, todėl parinkti artimiausi atitikmenys)

1.2. Komponentų pagrindimas

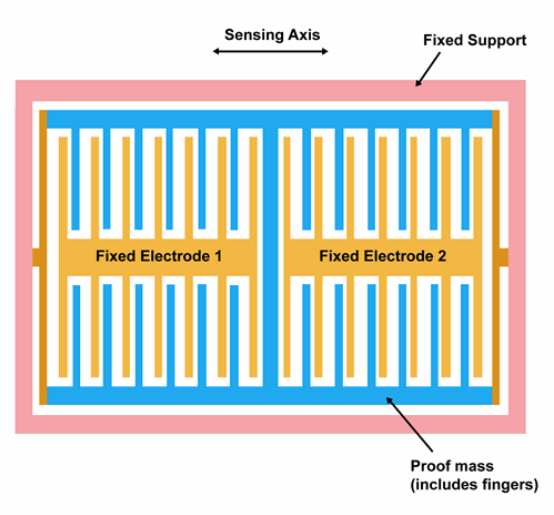
STM32F303VC mikrokontroleris ant DISCOVERY plokštės kartu integruoja ir sensorių modulį LSM303DLHC. Mikrokontroleris turi daug atminties konfigūracijos išsaugojimui (256kB FLASH), taip pat kelis I2C kanalus bendravimui su sensoriumi, ekranu bei UART kanalus komunikacijoms su kompiuteriu. Magnetinio lauko stiprumą ir žemės sunkio vektorių matuosime integruoto grandyno sensoriaus pagalba. Šis LSM303DLHC modulis inkorporuoja du sensorius:

1. 3D magnetometras - matuoja +- 8.1 Gauss magnetinio stiprumo laukus. Žemės magnetinio lauko stiprumas skirtingose vietose kinta tarp 0.25 ir 0.65G [6], todėl sensoriaus diapazono tikrai užteks. Šis sensorius veikia holo efekto principu, matuoja magnetinį lauką netiesiogiai: Tekanti srovė per plokštelę reaguoja į erdvės magnetinį lauką. Jis veikia elektronus Lorenco jėga ir stumia į vieną ar kitą pusę. Šių elektronų kelias išsikreipia nuo tiesės, susidaro krūvių skirtumas tarp sienelių ir iš to kylantis įtampos pokytis parodo magnetinio lauko stiprį skirtingų ašių atžvilgiu.



3 pav. Holo efekto įtaka elektronų judėjimui

2. 3D akselerometras – matuoja +-16g pagreitį. Šis sensorius paremtas MEMS technologija, pagreitį matuoja netiesiogiai: Integruotuose grandynuose ant silicio sluoksnio yra suformuojamos spyruoklės laikančios didesnę masę. Ši masė vykstant sensoriaus judėjimui suspaudžia atitinkamą spyruoklę ir priartėja prie jos sienelės. Šis tarpas tarp sienelių sudaro kondensatorių, kurio kintamą talpą išmatavus gaunama dabartinio pagreičio reikšmė. Nustatome matavimų diapazoną į mažiausią galimą: +-2g, nes matuosime žemės sunkio jėgos vektorių, kuris yra 9.8m/s^2 = 1g.



4 pav. MEMS akselerometro veikimo principas

Šie du sensoriai perduoda informaciją tuo pačiu I2C kanalu, tuo pačiu adresu tik skirtingais registrais, taip palengvinant duomenų nuskaitymą ir apdorojimą. Šie dviejų sensorių duomenys bus apdorojami sudėtingų matematinių algoritmų, kad būtų apskaičiuota tikroji magnetinės šiaurės kryptis. Du sensoriai naudojami atkompensuoti nehorizontalų sensoriaus laikymą, vieno magnetometro užtektų, jei įrenginys būtų visada laikomas horizontaliai, todėl norėdami išvengti vartotojo sukurtų problemų įdiegiame visapusį režimą.

Šių dviejų sensorių išpildymas ant tos pačios plokštės garantuoja gerą komunikaciją be išorinių triukšmų, gerą maitinimo prijungimą ir leidžia greitai prototipuoti sistemas nesiveliant į savo paties PCB plokštės dizaino problemas. Sensoriai veikia su 3.3V įtampos lygiais.

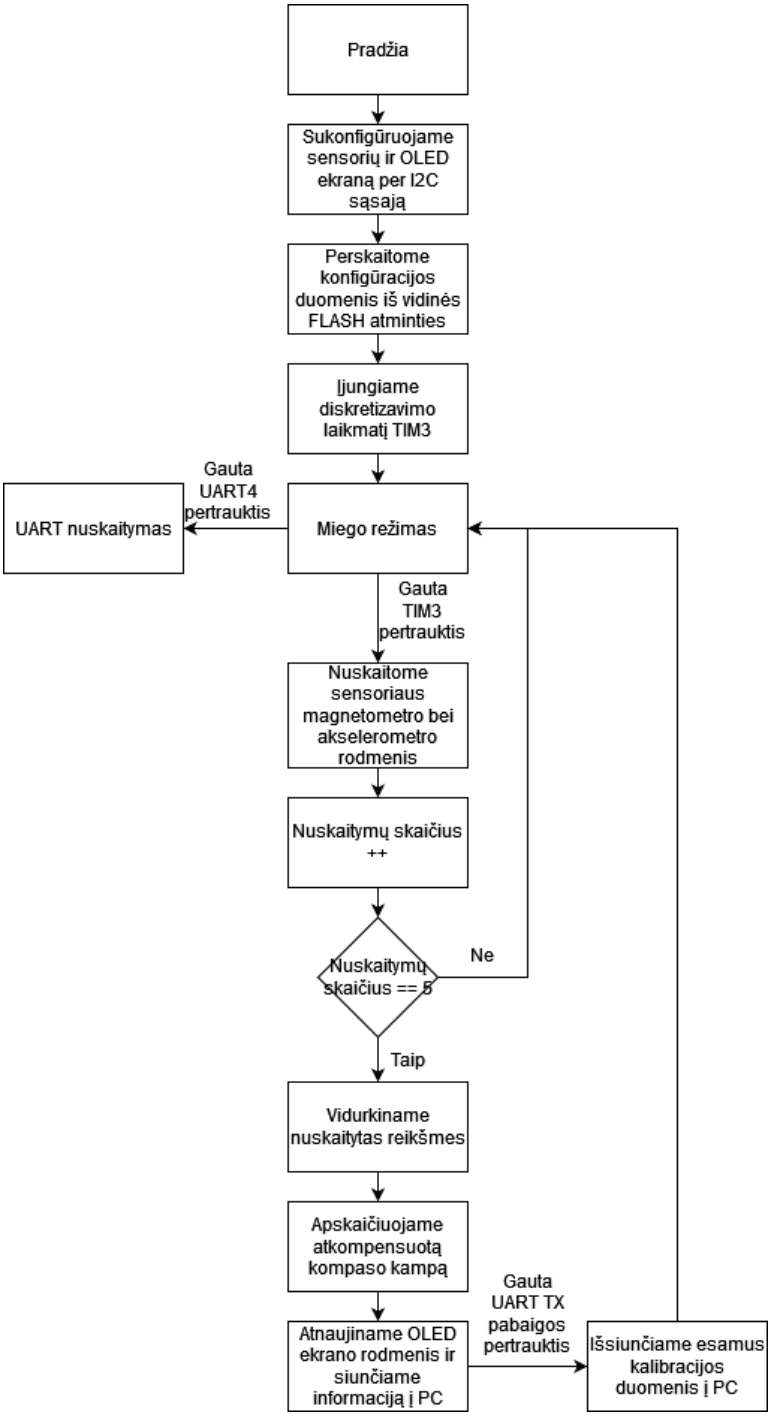
OLED ekranas turi pakankamai vietos atvaizduoti ir kompaso nuokrypio nuo šiaurės laipsnius, kurių bus daugiausia trys skaičiai, ir yra pakankamai vietos atvaizduoti grafiniam vaizdui – apvaliam kompasui su šiaurės rodykle. Taip pat maitinimas ir valdymo įtampos veikia 3.3V lygyje, kas yra suderinama su pasirinktu mikrovaldikliu.

UART į USB konverteris priima ir 5V ir 3.3V UART signalus, gamintojas turi veikiančius Windows draiverius bei garantuoja veikimą 300bps iki 1.5Mbps greičiais.

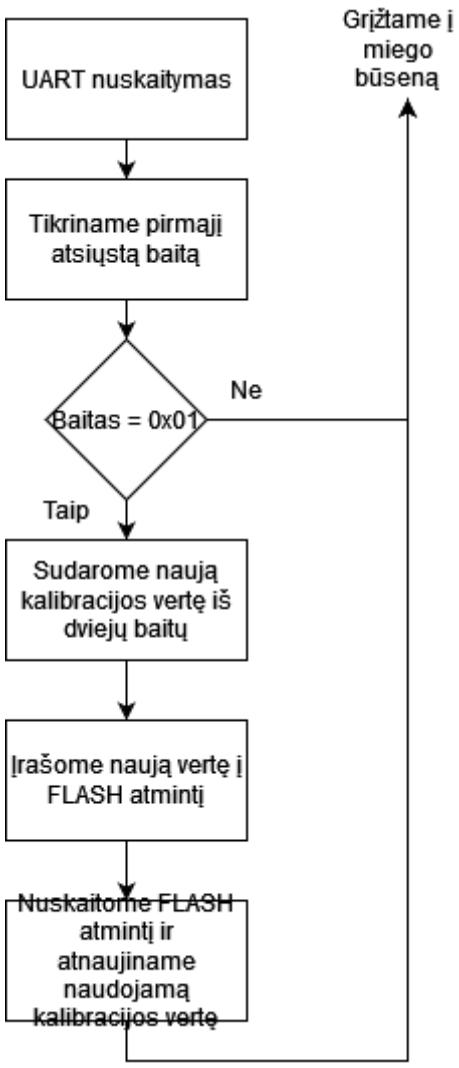
Mikrokontrolerio maketo plokštė maitinama ir programuojama per integruotą ST-Link USB B mini kabelio pagalba. Mikrovaldiklis veikia su 3.3V įtampa, bet į kai kuriuos išvadus gali priimti ir 5V signalus.

2. Programos schema

2.1. Algoritmo aprašas



5 pav. Mikrokontrolerio pagrindinės programos flow chart



6 pav. Programos konfigūravimo duomenų priėmimo iš kompiuterio flow chart tęsinys

2.2. Matematinis pagrindimas

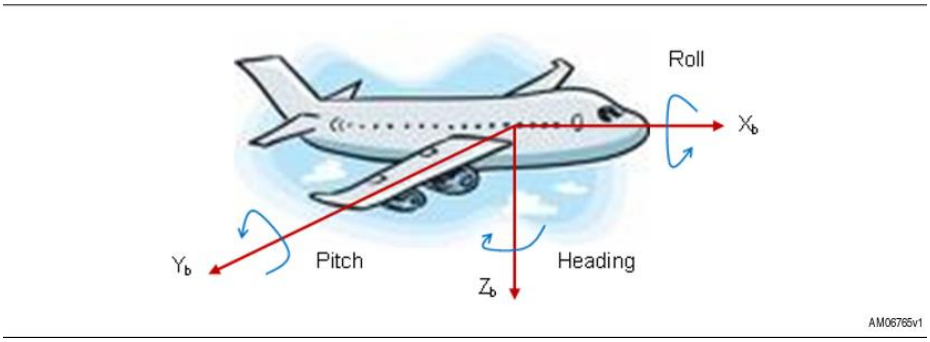
Laikant įrenginio plokštę nehorizontaliai sensoriaus matavimo ašys pasislenka ir rodo duomenis, kurie su įprasta apskaičiavimo formule nėra teisingai apdorojami, todėl kompaso nehorizontaliai padėčiai atkompensuoti reikia panaudoti duomenis iš dviejų sensorių:

1. 3D Akselerometro – žemės sunkio vektoriaus atradimui ir įrenginio pasukimo apskaičiavimui
2. 3D Magnetometro – magnetinių laukų stiprumui rasti ir žinant įrenginio pasukimą apskaičiuoti šiaurės krypties vektorių

Abu sensoriai yra integruoti į vieną komponentą.

2.2.1. Įprastas šiaurės krypties skaičiavimo metodas

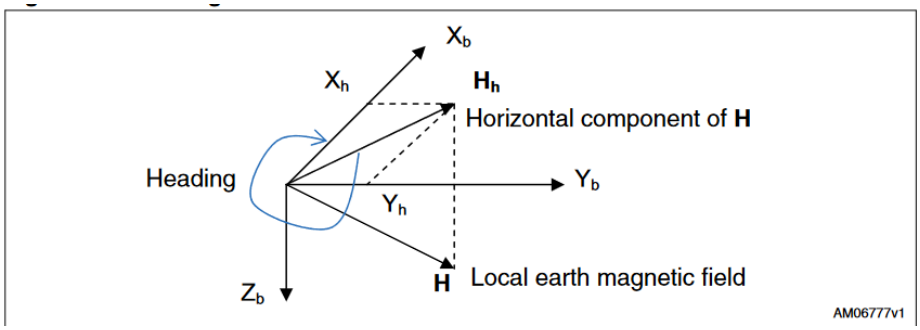
Objekto pasisukimą erdvėje galima nusakyti trimis dydžiais: roll, pitch ir heading.



7 pav. Objekto orientacijos iliustracija

Jei įrenginys yra horizontalioje būsenoje (pitch = roll = 0), tai magnetinio lauko komponentę (rodančią šiaurės krypties link) H galima suprojektuoti į horizontalią plokštumą, randant Hh – kryptį magnetinės šiaurės link. Pasinaudojant magnetometro X ir Y komponentėmis, arktangento funkcijos pagalba galima rasti šį vektorių.

$$\text{Heading} = \arctan(Y_h / X_h) \quad (1)$$



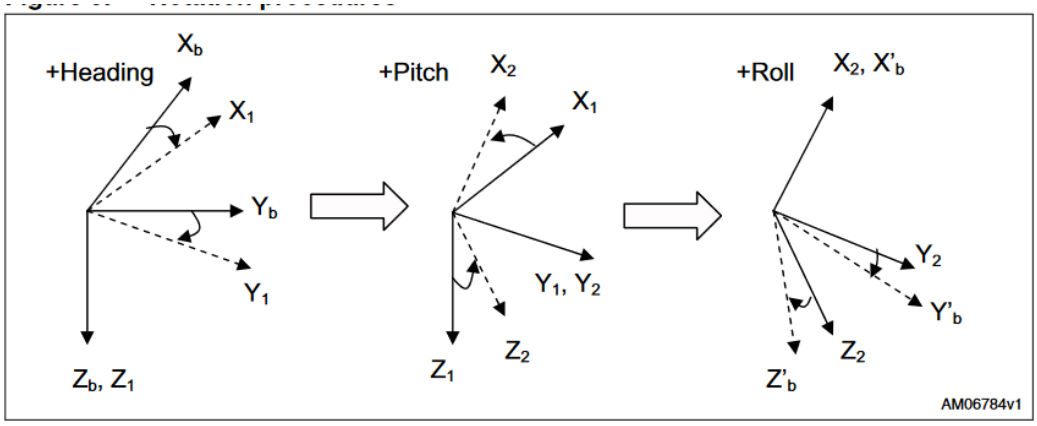
8 pav. Šiaurės krypties (heading) radimo iliustracija

2.2.2. Atkompensuotos šiaurės krypties skaičiavimo metodas

Kompaso modulį laikant ne horizontalioje padėtyje sensoriaus X ir Y rodmenys išsikreipia ir reikia įskaičiuoti ir Z ašies duomenis.

Pirma turime apskaičiuoti įrenginio pitch ir roll vertes pasitelkus akselerometro rezultatais:

Ieškant šių verčių turime susidaryti pasukimo matricas iš kurių galėsime atrasti tikrąsias plokštės posūkio komponentes.
Sudarant pasukimo matricas mūsų objektas turi būti pirma pasukamas aplink Z ašį kampu ψ, tada Y kampu ρ, X kampu γ:



9 pav. Objekto pasukimo erdvėje transformacijos ieškant pasukimo kampų

Iš šių pasukimų ir jų kampų aprašomos pasisukimų matricos gauname originalaus objekto pasukimų kampus Xb, Yb, Zb paverstus į horizontaliai sulygiuotas X‘b, Y‘b, Z‘b:

$$R_{\psi} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{\rho} = \begin{bmatrix} \cos\rho & 0 & -\sin\rho \\ 0 & 1 & 0 \\ \sin\rho & 0 & \cos\rho \end{bmatrix}$$

$$R_{\gamma} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix}$$

$$\begin{bmatrix} X'_b \\ Y'_b \\ Z'_b \end{bmatrix} = R_{\gamma} R_{\rho} R_{\psi} \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix}$$

$$= \begin{bmatrix} \cos\rho\cos\psi & \cos\rho\sin\psi & -\sin\rho \\ \cos\psi\sin\rho\sin\gamma - \cos\gamma\sin\psi & \cos\gamma\cos\psi + \sin\rho\sin\gamma\sin\psi & \cos\rho\sin\gamma \\ \cos\psi\sin\rho\cos\gamma + \sin\gamma\sin\psi & -\sin\gamma\cos\psi + \sin\rho\cos\gamma\sin\psi & \cos\rho\cos\gamma \end{bmatrix} \cdot \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} \quad (2)$$

Ši formulė mums leidžia žinant esamas ir norimas gauti koordinates laisvai pasukti bet kokį kūną. Kadangi horizontaliai sulygiuotos plokštės Xb ir Yb ašių pasisukimai bus lygūs 0, tai įvedus akselerometro duomenis Ax1, Ay1, Az1 gaunama formulė:

$$\begin{bmatrix} A_{x1} \\ A_{y1} \\ A_{z1} \end{bmatrix} = \begin{bmatrix} \cos\rho\cos\psi & \cos\rho\sin\psi & -\sin\rho \\ \cos\psi\sin\rho\sin\gamma - \cos\gamma\sin\psi & \cos\gamma\cos\psi + \sin\rho\sin\gamma\sin\psi & \cos\rho\sin\gamma \\ \cos\psi\sin\rho\cos\gamma + \sin\gamma\sin\psi & -\sin\gamma\cos\psi + \sin\rho\cos\gamma\sin\psi & \cos\rho\cos\gamma \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (3)$$

Ji supaprastinama iki:

$$\begin{aligned} \text{Pitch} &= \rho = \arcsin(-A_{x1}) \\ \text{Roll} &= \gamma = \arcsin(A_{y1} / \cos\rho) \end{aligned} \quad (4)$$

Gavę dvi mums reikiamas pasisukimo vertes galime jas įvertinti skaičiuodami atkompensuotą magnetinio sensoriaus šiaurės kryptį:
Imdami (3) formulės matricą ją galime panaudoti magnetinio lauko pasukimui rasti ir ją išskleidę gausime atkompensuoto magnetinio lauko komponentes (Mx2, My2, Mz2) iš nuskaitytų sensoriumi (Mx1, Mx1, Mz1).

$$\begin{aligned} M_{x2} &= M_{x1}\cos\rho + M_{z1}\sin\rho \\ M_{y2} &= M_{x1}\sin\gamma\sin\rho + M_{y1}\cos\gamma - M_{z1}\sin\gamma\cos\rho \\ M_{z2} &= -M_{x1}\cos\gamma\sin\rho + M_{y1}\sin\gamma + M_{z1}\cos\gamma\cos\rho \end{aligned} \quad (5)$$

Įstatę jau apskaičiuotas pitch ir roll reikšmes ir panaudoję pradinę (1) formulę gausime atsakymą [3].

2.3. Vartotojo kalibracijos aprašymas

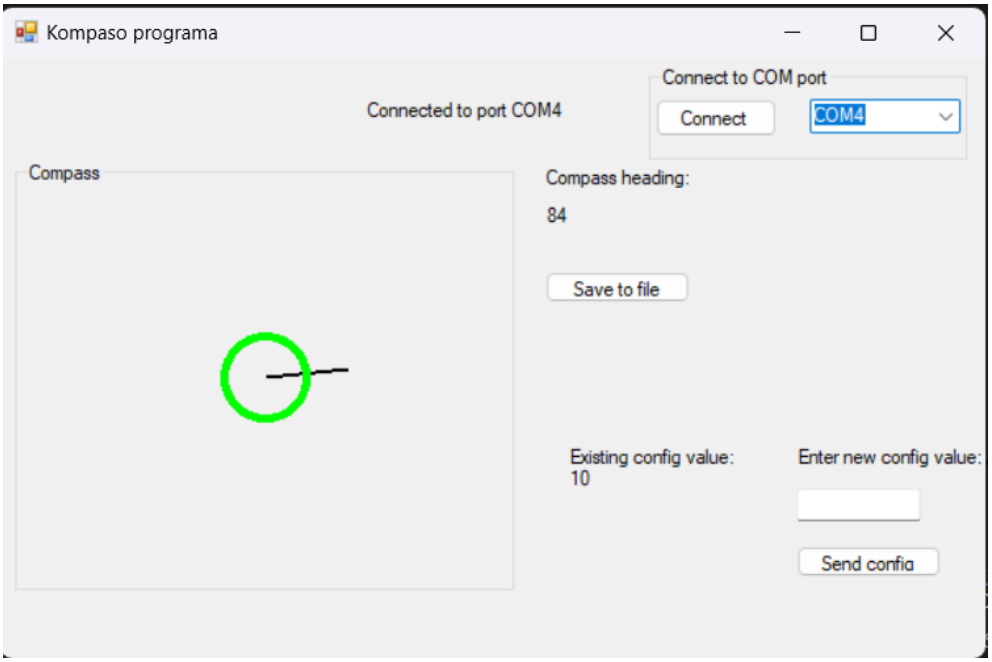
Nors sensorius yra individualiai sukalibruotas gamykloje, bet papildomų netikslumų gali atsirasti einant laikui arba norint tiesiog pakeisti rodmenis (pridėti deklinacijos vertę atsižvelgiant į esamą įrenginio platumą, jei norima, kad kompasas rodytų geografinės šiaurės link). Norint labai paprastai tai įvykdyti, prie apskaičiuotų iš sensoriaus laipsnių galima pridėti konfigūracijos vertę. Ši vertė gali būti saugoma mikrokontrolerio vidinėje Flash tipo atmintyje, todėl bus išsaugota tarp įjungimų, ją nustatyti galima atsiuntus UART komandą iš kompiuterio programos.

Mikrokontrolerio Flash atmintis yra suskirstyta į 128 puslapius po 2kB. Kadangi atminties pradžioje ji yra skirta programai laikyti, pasirenkame paskutinį atminties puslapį adresu 0x0803F800, kuriame yra mažiausias šansas sugadinti esamus ar mūsų įrašytus duomenis [7].

Mikrokontroleryje atmintis laikoma po 32 bitų ilgio žodžius, todėl skaitysime ir rašysime tokio ilgio informaciją. Rašant į atmintį reikia išjungti jos apsaugas, tam padaryti panaudojame jau esamą internete prieinamą biblioteką [5].

2.4. Kompiuterinės programos galimybės

Personaliniam kompiuteriui parašyta programa naudojantis C# programavimo kalba, Windows Forms sistemos pagalba, per UART sąsają COM port ir „System.IO.Ports“ biblioteką geba perduoti ir priimti informaciją iš kompiuterio.

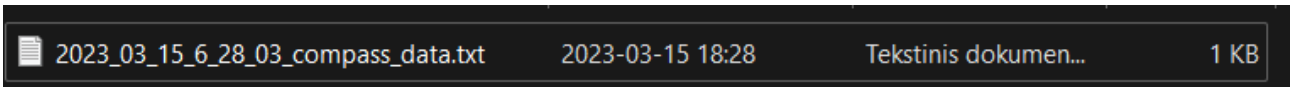


10 pav. Pavyzdinis kompiuterio programos vaizdas

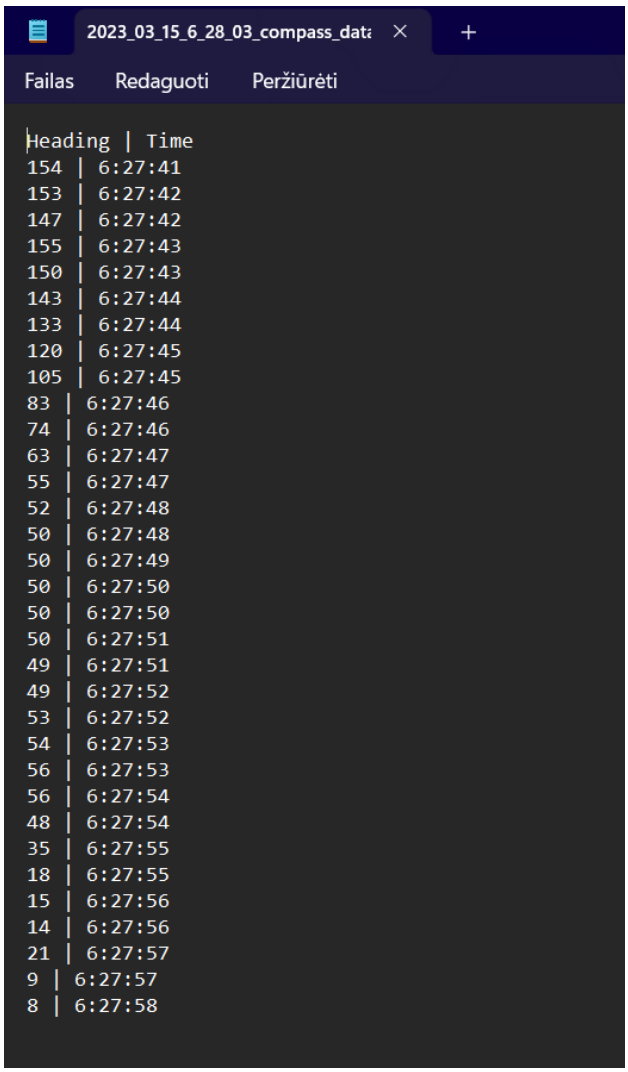
Ši programa sugeba:

- Prisijungti prie atitinkamo COM porto ir pranešti vartotojui apie prisijungimo stadiją
- Priimti kontrolerio informaciją ir ją atvaizduoti ekrane
- Siųsti į kontrolerį kalibravimo duomenis
- Išsaugoti surinktą informaciją tekstinio failo pavidalu

Priimant informaciją ji yra išsaugoma kartu su priėmimo laiko momento data. Vartotojui paspaudus “Save to file” mygtuką surinkti duomenys yra išsaugomi tekstinio failo pavidalu. Sukuriamas failas išsaugojimo datos pavadinimu o jame išsaugomi kompaso kampo duomenys su priėmimo laikų momentais.



11 pav. Išsaugotas pavyzdinis failas



12 pav. Pavyzdinio failo išsaugota informacija

2.5. Komunikacijų su kompiuteriu aprašas

Komunikacijos su kompiuteriu vykdomos per UART sąsają. Yra sukurtos trys komandos komunikacijai iš ir į kompiuterį.

Pirmasis siunčiamas baitas yra ID/Komandos numeris, kad priimantis įrenginys galėtų atpažinti kokie duomenys yra siunčiami. Likę du baitai skirti naudingai informacijai siųsti.
Pasirinkti du informacijos baitai, nes siunčiamos kampų ar konfigūracijos vertės yra 0 – 360 laipsnių diapazone ir vieno baito tam neužtektų. Galima buvo dalį informacijos įrašyti į komandos baitą, bet tai būtų apsunkinę nuskaitymą.

Lentelė 3. Komunikacijų komandų aprašas

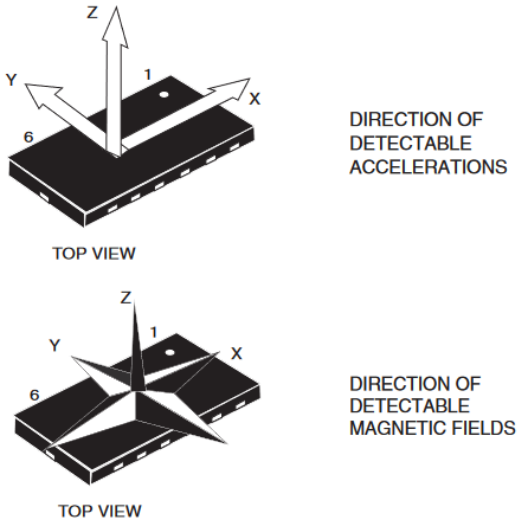
ID/Komandos numeris	Kryptis	Baitų kiekis	Aprašas
1	Kompiuteris -> kontroleris	3	Atnaujina mikrokontrolerio kalibracijos vertę, įrašo ją į jo vidinę atmintį.
3	Kontroleris -> kompiuteris	3	Nusiunčia esamo kampo nuo šiaurės reikšmę kompiuteriui atvaizduoti ekrane.

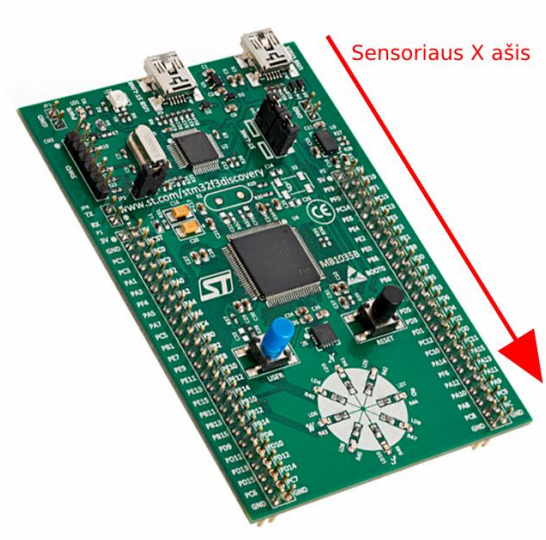
4	Kontroleris -> kompiuteris	3	Nusiunčia esamos kalibracijos reikšmę į kompiuterį.

3. Testavimas ir rezultatai

3.1. Testavimas

Kompaso X ašis yra nukreipta lygiagrečiai plokštės ilgio tolyn nuo USB jungties, ją laikysime kaip priekio kryptį. Palei ją skaičiuosime nuokrypio nuo šiaurės kampą.





13 pav. Sensoriaus ašys plokštės atžvilgiu

Kadangi neturime tikslaus būdo sukurti norimo stiprumo ir krypties magnetinį lauką realiame pasaulyje, tai ribinių verčių testavimą atliksime kodo simuliacijoje:

Kadangi iš sensoriaus gaunamos 16bitų vertės koduotos “two’s complement”, kas reiškia, jog saugomi skaičiai diapazone $\mathbf{[-2^{N-1}, 2^{N-1} - 1]}$, mūsų atveju: [−32,768; 32,767].
Todėl į rawX, rawY, rawZ rašysime neapdorotus magnetometro ašių baitus, o į accX, acccY, accZ – akselerometro ašių baitus.

Simuliavimo testavimo rezultatai:

Name	Value	Type
D	135	int
accX	0	short
accY	0	short
accZ	-32767	short
rawX	-32767	short
rawY	32767	short
rawZ	0	short
<Enter expression>		

14 pav. Paduodamos maksimalios vertės laikant horizontaliai (be kompensavimo)

Paveiksliuke matome, kad pagreitis nukreiptas Z ašimi žemyn, magnetinis laukas maksimalus neigiama X kryptimi (į vartotoją) ir teigiama Y ašim (į kairę). Išėjime matome kampo reikšmę D rodančią 135 laipsnius, kas matuojant prieš laikrodžio rodyklę yra pietvakariai. Rodo teisingai.

D	270	int
accX	0	short
accY	-32767	short
accZ	0	short
rawX	0	short
rawY	0	short
rawZ	-32767	short
<Enter expression>		

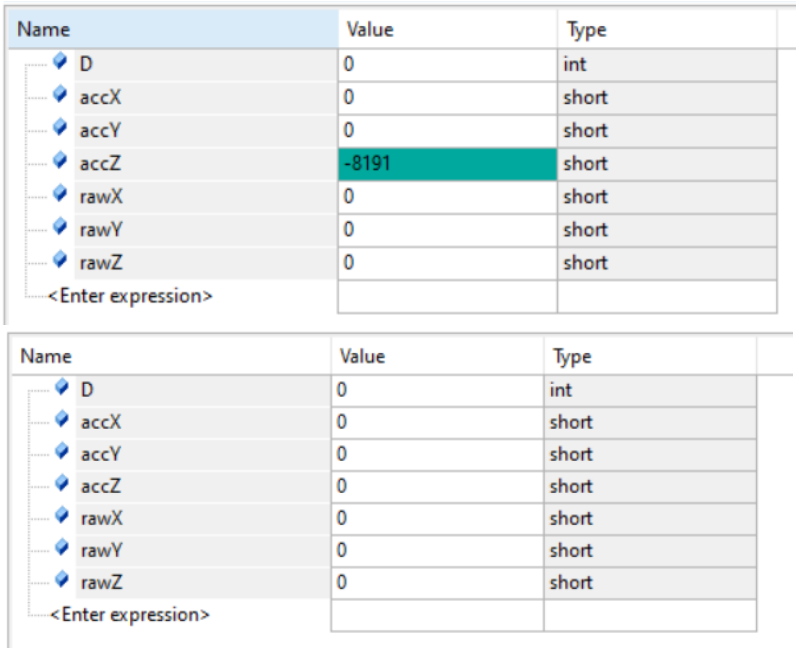
15 pav. Paduodamos maksimalios vertės laikant ne horizontaliai (su kompensavimu)

Kompasas paverstas ant dešiniojo šono, magnetinis laukas neigiama Z ašimi, kas yra žemyn nuo kompaso arba į kairę, o 270 laipsnių šiuo atveju rodo ta kryptimi (Šiuo atveju testavimas nekorektiškas, nes vakarai ir rytai gali būti interpretuojami kaip teisingos kryptys magnetiniam laukui žemyn parodyti).

Name	Value	Type
D	63	int
accX	0	short
accY	0	short
accZ	-16383	short
rawX	8191	short
rawY	16383	short
rawZ	0	short
<Enter expression>		

16 pav. Paduodamos įprastos vertės

Kompasas laikomas horizontaliai su silpnu magnetiniu lauku X kryptimi ir stipresniu Y, kas išeina į didesnį nei 45 laipsnių kampo pasisukimą į šiaurės vakarus.



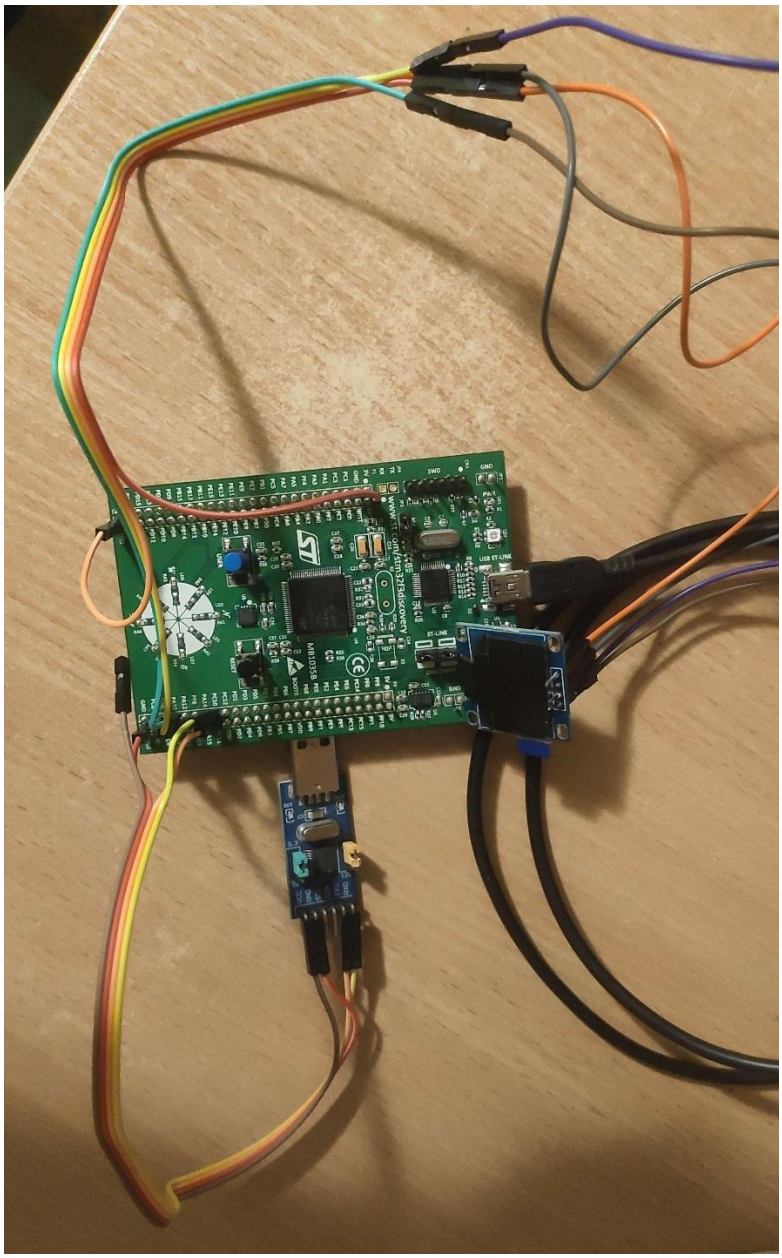
Name	Value	Type
D	0	int
accX	0	short
accY	0	short
accZ	-8191	short
rawX	0	short
rawY	0	short
rawZ	0	short
<Enter expression>		

Name	Value	Type
D	0	int
accX	0	short
accY	0	short
accZ	0	short
rawX	0	short
rawY	0	short
rawZ	0	short
<Enter expression>		

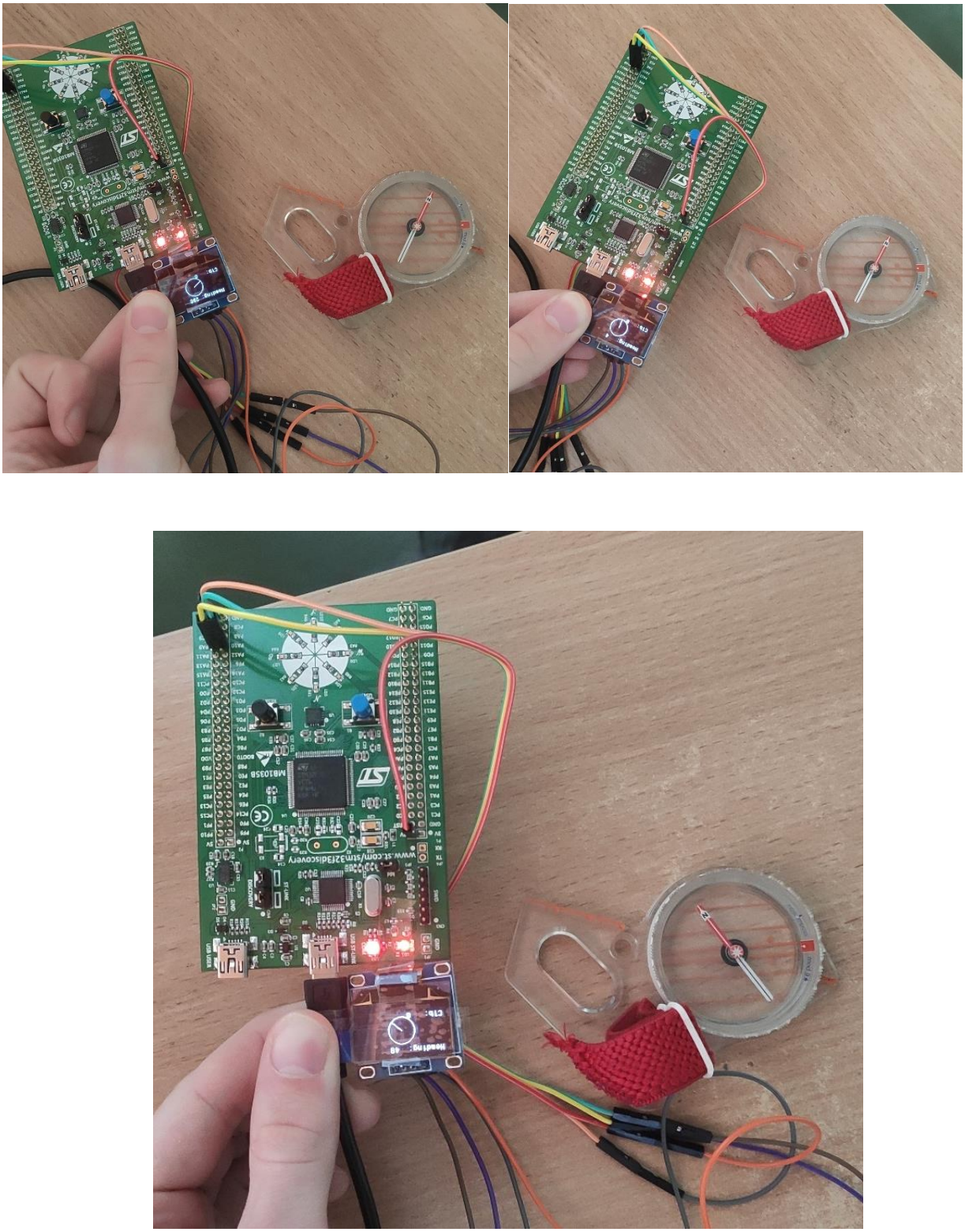
17 pav. Nesant duomenų nėra rodomos neaiškios vertės

Kaip matome, su visais variantais, visais pasukimais, visomis diapazono reikšmėmis įrenginys veikia gerai. Kampas gaunamas tikslus, nesant jokių iš sensoriaus paimamų duomenų (visur nuliai) išėjime irgi gaunamas nulis, tai parodo, kad skaičiavimai nėra atsitiktiniai ir nėra jokios statinės dedamosios.

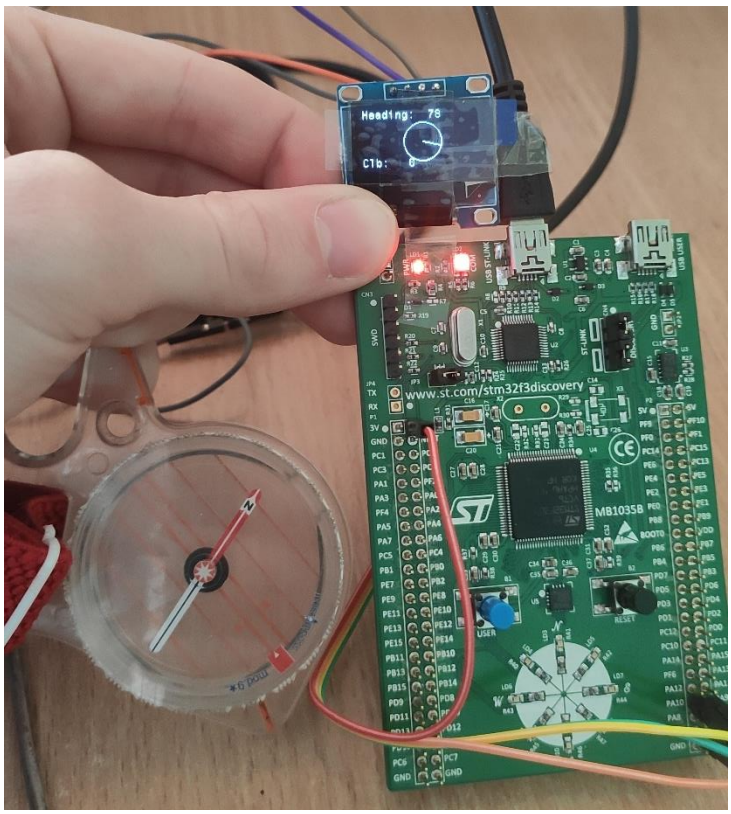
3.1.1. Fizinis veikimo testavimas



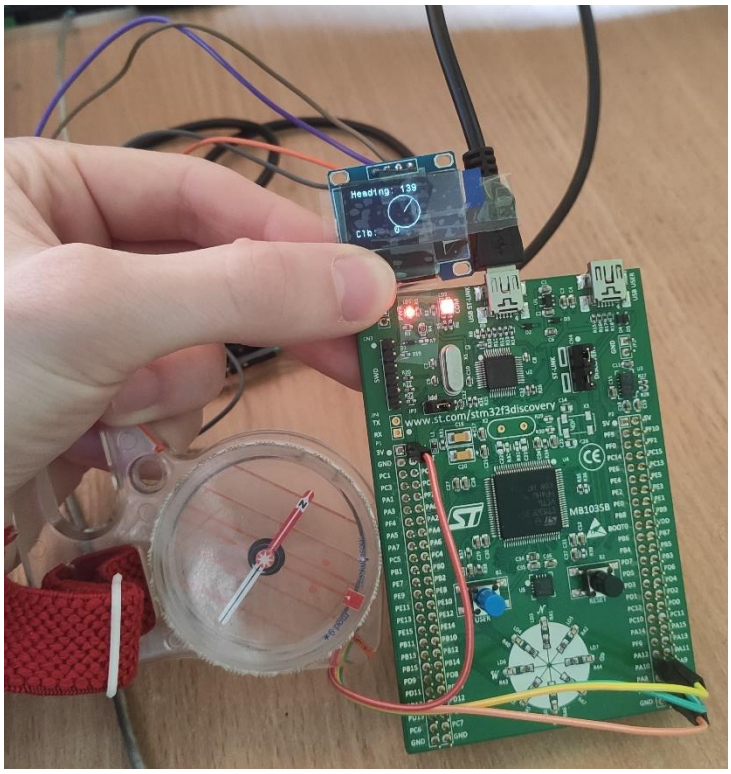
18 pav. Realaus pilno įtaiso sujungimas



19 pav. Kompaso ir mikrokontrolerio rodomo kampo fizinis testavimas, indikatoriaus rodoma kryptis atitinka realaus kompaso kryptį



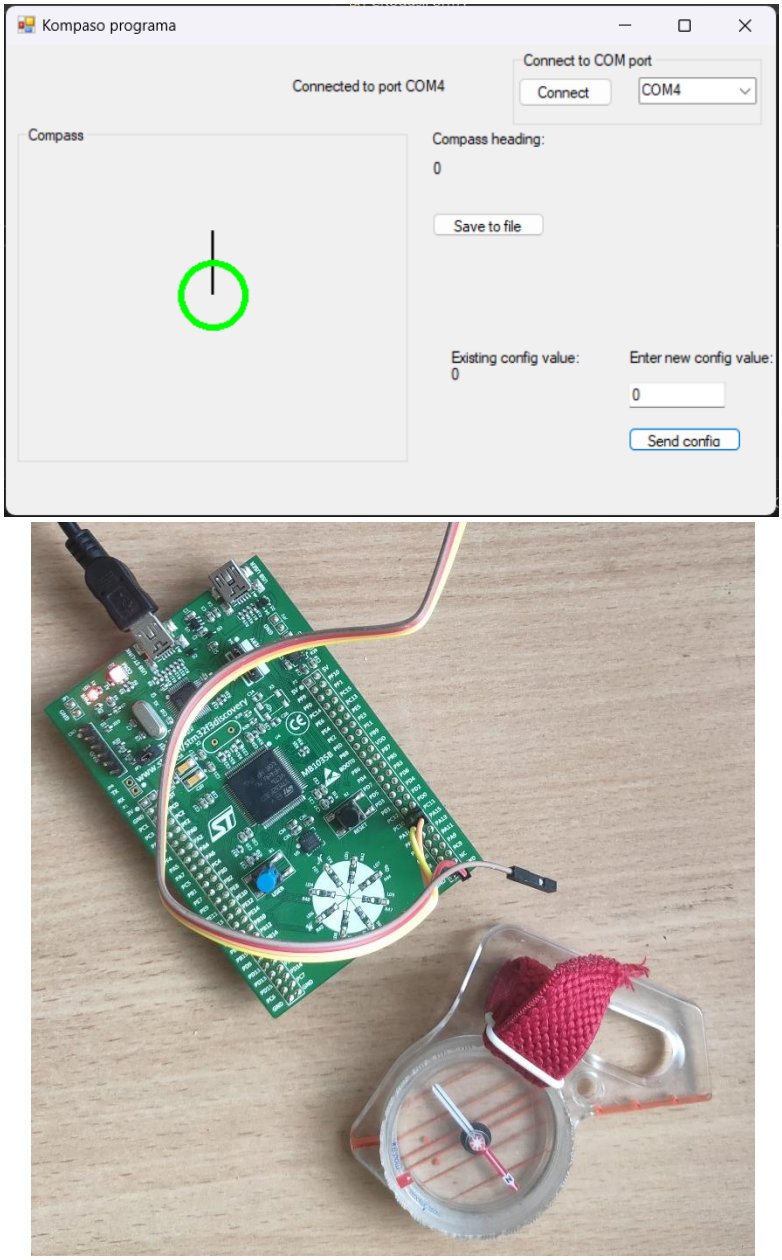
20 pav. Kampu laikomo kompaso vaizdas išjungus nehorizontalios laikymo padėties kompensavimą



21 pav. Kampu laikomo kompaso testavimas įjungus kompensavimą, matome, kad skaičiavimai veikia gerai, atitinka realaus kompaso rodmenis

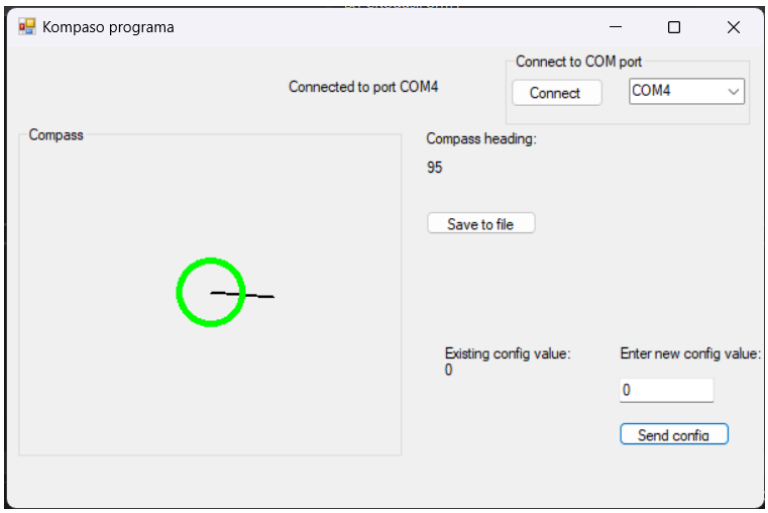
(Nuotraukose atjungtas UART-USB konverteris dėl patogesnio laikymo)

3.1.2. Kompiuterinės programos veikimo testavimas

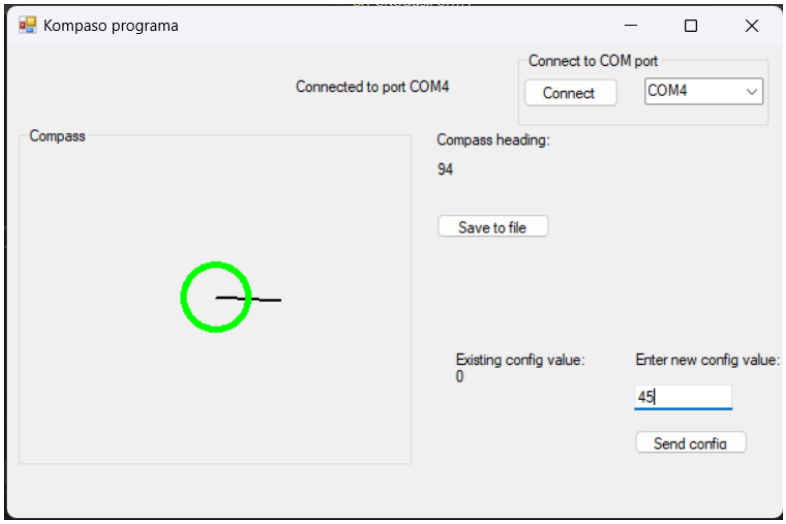


22 pav. Kompaso realybėje ir kompiuterinėje programoje vaizdas, kai sensorius nukreiptas šiaurės kryptimi

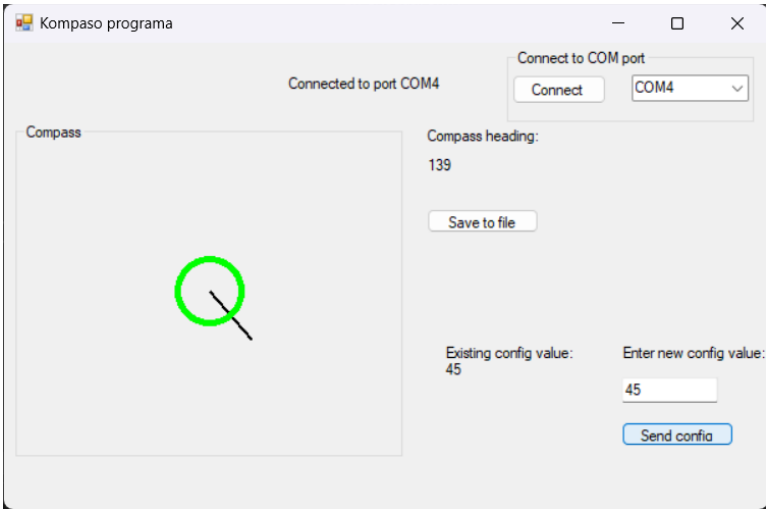
Norint įrašyti naują konfigūravimo reikšmę į mikrovaldiklį:



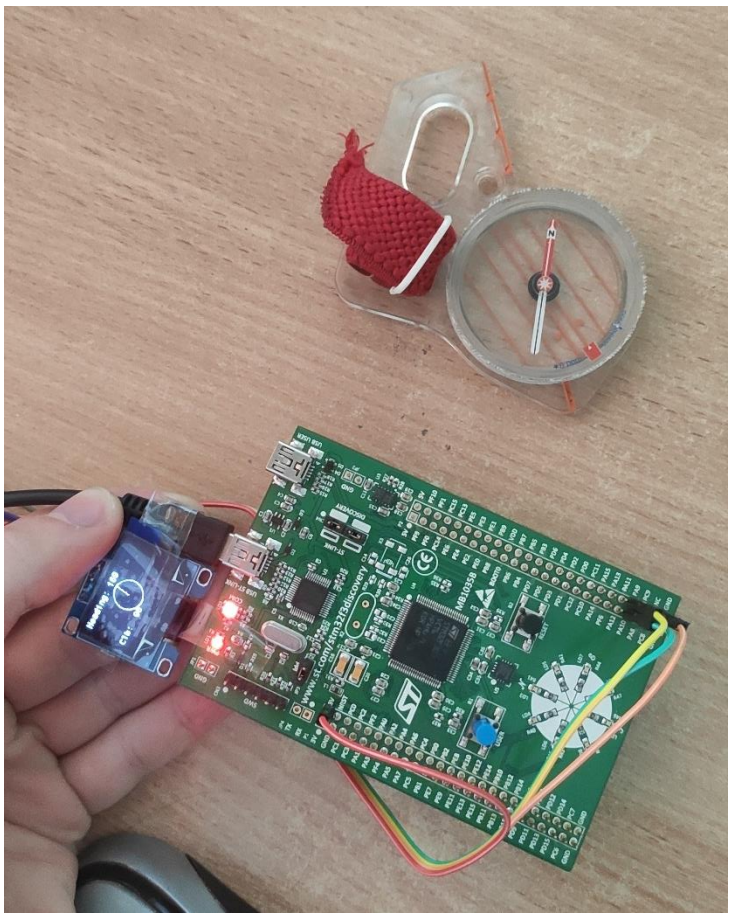
23 pav. Kompiuterio programos vaizdas prieš konfigūravimo reikšmės įrašymą



24 pav. Kompiuterio programos vaizdas prieš 45 laipsnių konfigūravimo vertės išsiuntimą (dešinė apačia)



25 pav. Kompiuterio programos vaizdas po naujos konfigūravimo vertės išsiuntimo nepakeitus fizinio sensoriaus kampo, matome, kad rodomas kampas pasisuko 45 laipsniais palei laikrodžio rodyklę



26 pav. 90 laipsnių įrašyta konfigūracijos reikšmė OLED ekrane

3.2. Paklaidų vertinimas

Ištestuoti paklaidas eksperimentiniu būdu nėra lengva, nes neturime ar tikslių kampo matavimo priemonių ar etaloninio magneto, kad ištestuoti sukurto magnetinio lauko stiprumą. Todėl paklaidas vertinsime tik teoriškai.

Sukurtos įterptinės sistemos paklaidos gali atsirasti aibėje vietų:

- Sensoriaus charakteristikos
- Aplinkos triukšmai
- Skaičių apvalinimas

3.2.1. Sensoriaus paklaidos

Sensoriaus matavimo paklaidos atsiranda iš ribotos sensoriaus rezoliucijos, sensoriaus charakteristikos ir kitų parametrų. Visos paklaidos pateikiamos sensoriaus duomenų lape:

Symbol	Parameter	Test conditions	Min.	Typ.(1)	Max.	Unit
LA_TCSo	Linear acceleration sensitivity change vs. temperature	FS bit set to 00		±0.01		%/°C
LA_TyOff	Linear acceleration typical Zero-*g* level offset accuracy(3),(4)	FS bit set to 00		±60		mg
LA_TCOff	Linear acceleration Zero-*g* level change vs. temperature	Max delta from 25 °C		±0.5		mg/°C
LA_An	Acceleration noise density	FS bit set to 00, normal mode(*Table 8.*), ODR bit set to 1001		220		ug/(√Hz)
M_R	Magnetic resolution			2		mgauss
M_CAS	Magnetic cross-axis sensitivity	Cross field = 0.5 gauss H applied = ±3 gauss		±1		%FS/ gauss
M_EF	Maximum exposed field	No permanent effect on sensor performance			10000	gauss
M_DF	Magnetic disturbance field	Sensitivity starts to degrade. Use S/R pulse to restore sensitivity			20	gauss
Top	Operating temperature range		-40		+85	°C

27 pav. Sensoriaus tikslumo charakteristikos

Konfigūravimo metu pasirinktas FS = b00, GN = b001.
Matome, kad akselerometro paklaidos priklauso nuo temperatūros, +-0.01%/°C, taip pat yra pastovus +-60mg nulio g offsetas.

Matavimo metu esant 20°C temperatūrai paklaida nuo rezoliucijos susidaro:

$$Paklaida = 0.01 * 20 = 0.2\%$$

Akselerometro rezoliucija 1mg, todėl:

$$Paklaida = 0.2\% * 0.001 = 2ug$$

Nuo nuskaitymo greičio kuris nustatytas į 400Hz kyla paklaida 220ug/sqrt(Hz):

$$Paklaida = \frac{220}{\sqrt{400}} = 11ug$$

Bendros didžiausios akselerometro paklaidos:
$P_{aksel} = 60*10^{-3} + 11*10^{-6} + 2*10^{-6} = 60{,}13*10^{-3}$g

Magnetometras turi 1% mg/gauss paklaidą, jo rezoliucija 2mGauss, todėl:

$$Paklaida = 0.001\% * 0.002 = 0.02uGauss$$

Bendra magnetometro paklaida:
$P_{mag} = 11*10^{-6} * 0.02*10^{-6} = 11.02*10^{-6}$ Gauss

3.2.2. Apvalinimo paklaidos

Skaičiuojant galutinį šiaurės kampą apvaliname gautą reikšmę iki sveiko skaičiaus, todėl apvalinimo procese atsiranda +- 0.5° absoliutinė paklaida.
Taip pat skaičiavimuose naudojami float tipo kintamieji nėra visiškai tikslūs, dažnai yra apvalinami į artimiausią reikšmę, nes negali visų atvaizduoti dėl riboto bitų skaičiaus.

3.2.3. Rezoliucijos paklaidos

Rezoliucija individualiai nustatyta sensoriaus konfigūravimo metu.

Symbol	Parameter	Test conditions	Min.	Typ.(1)	Max.	Unit
LA_FS	Linear acceleration measurement range(2)	FS bit set to 00		±2		*g*
		FS bit set to 01		±4		
		FS bit set to 10		±8		
		FS bit set to 11		±16		
M_FS	Magnetic measurement range	GN bits set to 001		±1.3		gauss
		GN bits set to 010		±1.9		
		GN bits set to 011		±2.5		
		GN bits set to 100		±4.0		
		GN bits set to 101		±4.7		
		GN bits set to 110		±5.6		
		GN bits set to 111		±8.1		
LA_So	Linear acceleration sensitivity	FS bit set to 00		1		mg/LSB
		FS bit set to 01		2		
		FS bit set to 10		4		
		FS bit set to 11		12		
M_GN	Magnetic gain setting	GN bits set to 001 (X,Y)		1100		LSB/ gauss
		GN bits set to 001 (Z)		980		
		GN bits set to 010 (X,Y)		855		
		GN bits set to 010 (Z)		760		
		GN bits set to 011 (X,Y)		670		
		GN bits set to 011 (Z)		600		
		GN bits set to 100 (X,Y)		450		
		GN bits set to 100 (Z)		400		
		GN bits set to 101 (X,Y)		400		
		GN bits set to 101 (Z)		355		
		GN bits set to 110 (X,Y)		330		
		GN bits set to 110 (Z)		295		
		GN bits set to 111(2) (X,Y)		230		
		GN bits set to 111(2) (Z)		205		

28 pav. Sensoriaus rezoliucijos charakteristikos

Akselerometro rezoliucija: 1mg/LSB
Magnetometro rezoliucija X, Y ašys: 1100LSB/Gauss = 1/1100 = 0.9mGauss/LSB
Magnetometro rezoliucija Z ašis: 980LSB/Gauss = 1/980 = 1.02mGauss/LSB

Rezoliucijos paklaidos kyla iš apvalinimo, gavus magnetinę ar pagreičio reikšmę viduriuke tarp dviejų reikšmių vartotojui yra atiduodama arčiausiai esanti, todėl maksimali paklaida gaunama pusė rezoliucijos reikšmės.
Akselerometro: +-0.5mg
Magnetometro X, Y ašys: +-0.45mGauss
Magnetometro Z ašis: +-0.501mGauss

Geresniam tikslumui pasiekti sensoriaus LA_So ir LA_TyOff vertės yra individualiai sukalibruotos gamykloje, todėl individualaus kalibravimo atlikti nereikia.

Išvados

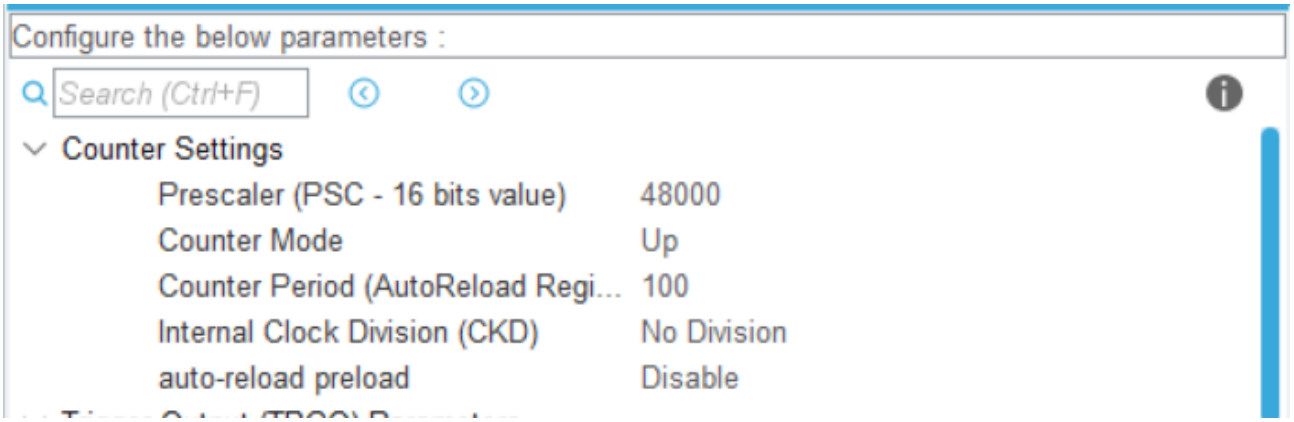
1. Sudarę įrenginio schemą, apjungę sensorių, ekraną ir USB-TTL modulį, suprogramavę vartotojo bei mikrokontrolerio programas sukūrėme savo, elektroninio kompaso duomenų surinkimo bei perdavimo įterptinę sistemą.
2. Sukurta kompiuterio programa sugeba priimdama informaciją ją išsaugoti tekstinio failo pavidalu su laiko momentais. Turint papildomų greičio sensoriaus duomenų apskaičiavus kiek laiko, kokiu vidutiniu greičiu, kokia kryptimi keliavo objektas galima būtų atkurti jo kelionės maršrutą ir dabartinę poziciją 3D erdvėje, todėl toks įrenginys galėtų būti naudojamas transporto priemonių navigacijai, jų sekimui ar lėktuvų juodosiose dėžėse.
3. Konfigūravimo vertė kuri pakeičia galutinį apskaičiuotą kampo reikšmę gali būti skirta arba kompaso rodymo paklaidoms kompensuoti (dirbant stipriai pašalinių magnetinių laukų veikiamose teritorijose), arba magnetinės šiaurės rodmenis pakeisti geografinės šiaurės rodmenimis. Deklinacijos kampas prie tam tikros platumos rodo kampo skirtumą tarp geografinės ir magnetinės šiaurės, jį įrašę į konfigūravimo reikšmę galime iš magnetinės šiaurės kompaso paversti jį į geografinės šiaurės kompasą.
4. Įrenginio kūrimo metu ilgai trukau, kol išsiaiškinau kaip veikia kompasas, visi duomenų nuskaitymai ir apdorojimai tikrai buvo nelengvi. Vietomis, prie tam tikrų situacijų kompaso rodmenys trumpam susimėto ir pradeda rodyti nesąmones, bet išsiaiškinti jų kilmės nesugebėjau.

Šaltiniai

1. https://www.st.com/en/evaluation-tools/stm32f3discovery.html#documentation
2. https://www.st.com/en/mems-and-sensors/lsm303dlhc.html
3. https://www.sparkfun.com/datasheets/Sensors/Magneto/Tilt%20Compensated%20Compass.pdf
4. https://controllerstech.com/oled-display-using-i2c-stm32/
5. https://controllerstech.com/flash-programming-in-stm32/
6. https://www.ncei.noaa.gov/products/geomagnetism-frequently-asked-questions
7. STM32 F3 reference manual

Priedai

1 Mikrokontrolerio kodas



29 pav. TIM3 konfigūravimo vaizdas

main.c failas

```
int compassRotation = 0;
int nuskaitymuSkaicius = 0;
//Gauta nuskaitymo taimerio pertrauktis
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
{
    HAL_ResumeTick();
    if(htim == &htim3){
        Get_Compass_Data();
        nuskaitymuSkaicius++;
        if(nuskaitymuSkaicius == 5){ //patikriname ar atnaujinti informacija
            #if Enable_sensor
            compassRotation = Calculate_compass_data();
            #endif
            #if Enable_screen
            Update_Screen();
            #endif
            #if Enable_pc
            Update_PC_Heading();
            #endif
            nuskaitymuSkaicius = 0;
        }

    }
}


//Gauta UART RX pertrauktis
//https://controllerstech.com/flash-programming-in-stm32/
uint8_t receiveUARTData[6];
uint16_t received_calibration = 0x0000;
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    HAL_ResumeTick();
    switch(receiveUARTData[0])
    {
        case 0x01: //GAvome nauja kalibracijos verte
            received_calibration = receiveUARTData[1] | (receiveUARTData[2] <<
8);
            Update_Calibration(received_calibration);
            Read_Calibration();
            break;
    }
}

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
   Update_PC_Calibration();
}
```

```
/* USER CODE BEGIN 2 */
   I2C_Init(); //Sukonfiguruojame sensoriu

   #if Enable_calibration
       Read_Calibration(); //perskaitome kalibracija irasyta FLASH atmintyje
   #endif

   #if Enable_screen //Ijungiame ekrana
       SSD1306_Init ();
       SSD1306_UpdateScreen();
   #endif
   HAL_TIM_Base_Start_IT(&htim3); //Pradedame nuskaitymo laikmati
 /* USER CODE END 2 */

 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
       HAL_UART_Receive_IT(&huart4, receiveUARTData, 3);
       HAL_SuspendTick();
   HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON,PWR_SLEEPENTRY_WFI);
       HAL_ResumeTick();
   /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
 }
```

mano.h failas

```
#define Enable_screen 1
#define Enable_sensor 1
#define Enable_pc 1
#define Enable_calibration 1

#define I2C_Compass_Adress 0x3C
#define I2C_Accelerometer_Adress 0x32
#define screen_width 128
#define screen_height 64

#define lsb_Gauss_xy 1100
#define lsb_Gauss_z 980
#define acc_full_scale 2.0f //Akselerometro duomenu pilnas diapozonas +-2g

extern I2C_HandleTypeDef hi2c1;
extern UART_HandleTypeDef huart4;

extern int compassRotation;
```

mano.c failas

```
uint8_t sendData[3];
uint8_t receiveData[6];
uint8_t status;

int calibrationValue = 0;
int D = 0;
//Sensoriaus inicializacija
void I2C_Init(){
    sendData[0] = 0x20;
    sendData[1] = 0x47; //50Hz akselerometro atnaujinimas, ijungimas asiu ir
sensoriaus
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Accelerometer_Adress,
(uint8_t *)sendData, 2, 100);

    sendData[0] = 0x23;
    sendData[1] = 0x08; //LSB lower ir High resolution +-2g
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Accelerometer_Adress,
(uint8_t *)sendData, 2, 100);

    sendData[0] = 0x23;
    sendData[1] = 0x90; //Filtrai akselerometro
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Accelerometer_Adress,
(uint8_t *)sendData, 2, 100);

    sendData[0] = 0x22;
    sendData[1] = 0x00; //Filtrai akselerometro
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Accelerometer_Adress,
(uint8_t *)sendData, 2, 100);

    //Ijungti magnetini kompasa
    sendData[0] = 0x00;
    sendData[1] = 0x14; //Duomenu atnajunimas 30Hz
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Compass_Adress, (uint8_t
*)sendData, 2, 100);

    sendData[0] = 0x02;
    sendData[1] = 0x00; //Pastoviai konvertuoti duomenis
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Compass_Adress, (uint8_t
*)sendData, 2, 100);

    sendData[0] = 0x01;
    sendData[1] = 0x80; //Diapozono nustatymas +-1.3Gauss
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Compass_Adress, (uint8_t
*)sendData, 2, 100);
}
```

```
void Get_Compass_Data()
{
    //Skaityti accelerometra
    sendData[0] = 0xA8;
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Accelerometer_Adress,
(uint8_t *)sendData, 1, 1);
    status = HAL_I2C_Master_Receive(&hi2c1, I2C_Accelerometer_Adress, (uint8_t
*)receiveData, 6, 1);
    accX_temp = receiveData[0] | (receiveData[1] << 8);
    accY_temp = receiveData[2] | (receiveData[3] << 8);
    accZ_temp = receiveData[4] | (receiveData[5] << 8);

    //Gauti magnetinius duomenis
    sendData[0] = 0x03;
    status = HAL_I2C_Master_Transmit(&hi2c1, I2C_Compass_Adress, (uint8_t
*)sendData, 1, 1);
    status = HAL_I2C_Master_Receive(&hi2c1, I2C_Compass_Adress, (uint8_t
*)receiveData, 6, 1);
    rawX_temp = (receiveData[0] << 8) | receiveData[1]; //Pasisukimas i sona
    rawZ_temp = (receiveData[2] << 8) | receiveData[3]; //Pasisukimas is
virsaus
    rawY_temp = (receiveData[4] << 8) | receiveData[5]; //persivertimas

    //Prideti vidurkio skaiciavimui
    nuskaitymu_skaicius++;

    rawX += rawX_temp;
    rawY += rawY_temp;
    rawZ += rawZ_temp;

    accX += accX_temp;
    accY += accY_temp;
    accZ += accZ_temp;
}
```

```
int Calculate_compass_data(){
    //Randame vidurki
    rawX = rawX / nuskaitymu_skaicius;
    rawY = rawY / nuskaitymu_skaicius;
    rawZ = rawZ / nuskaitymu_skaicius;

    accX = accX / nuskaitymu_skaicius/100.0f;
    accY = accY / nuskaitymu_skaicius/100.0f;
    accZ = accZ / nuskaitymu_skaicius/100.0f;
    accX = accX/64.0f/100.0f;
    accY = -accY/64.0f/100.0f;
    accZ = accZ/64.0f/100.0f;

    //Pasiverciame magnetine reiksme i gausus nuo 0 iki 1
    MX1 = rawX / (float)lsb_Gauss_xy; //lsb_Gauss_xy = 1100
    MY1 = rawY / (float)lsb_Gauss_xy;
    MZ1 = rawZ / (float)lsb_Gauss_z; //lsb_Gauss_z = 980


    //Skaiciuojame formules
    pitch = asin(-((float)accX/ 32768.0f/acc_full_scale)); //acc_full_scale =
2
    roll = asin((float)accY/ 32768.0f/acc_full_scale/cos((float)pitch));

    MX2 = MX1 * cos(pitch) + MZ1 * sin(pitch);
    MY2 = MX1 * sin(roll) * sin(pitch) + MY1 * cos(roll) -
MZ1*sin(roll)*cos(pitch);

    D = floor( atan2f(MY2, MX2) * 180.0f/3.14f); //Su atkompensavimais
    #if Enable_calibration

    D += calibrationValue;
    D %= 360;
    #endif
    if(D < 0){
        D += 360;
    }

    //Graziname reiksmes i 0
    nuskaitymu_skaicius = 0;
    rawX = 0;
    rawY = 0;
    rawZ = 0;
    accX = 0;
    accY = 0;
    accZ = 0;
    return D;
}
```

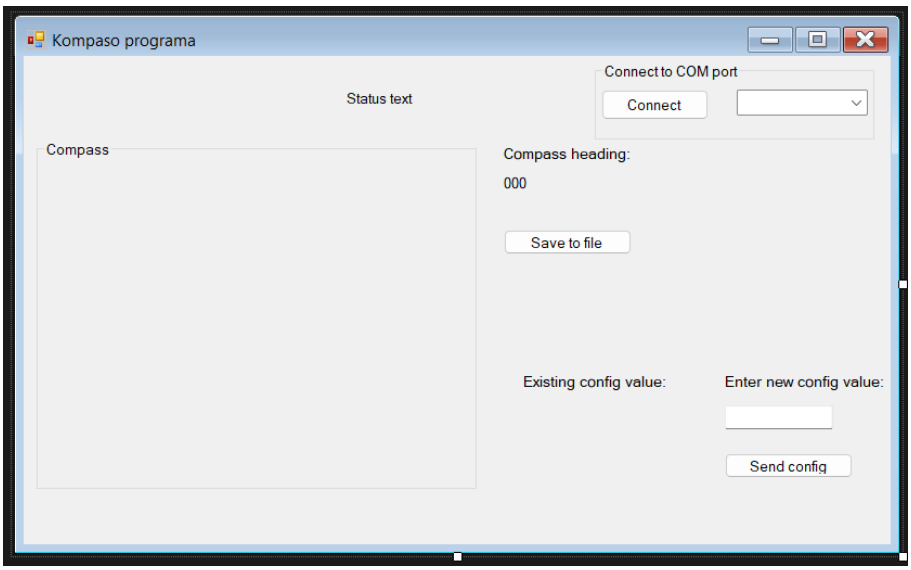
```
void Update_Screen(){
    SSD1306_Clear();
    //Irasyti pasisukimo reiksme ekrane
    char headingString[13];
    sprintf(headingString, "Heading: %3d", compassRotation);
    SSD1306_GotoXY (0,0);
    SSD1306_Puts (headingString, &Font_7x10, 1);
    //Irasyti konfiguracimo reiksme ekrane
    char calibrationString[9];
    sprintf(calibrationString, "Clb: %3d", calibrationValue);
    SSD1306_GotoXY (0,50);
    SSD1306_Puts (calibrationString, &Font_7x10, 1);

    //Nupiesti komapasa
    SSD1306_DrawCircle(screen_width/2, screen_height/2, 20,1);
    int comppasLineX = sin((float)compassRotation * 3.14f / (float)180) *
20.0;
    int comppasLineY = cos((float)compassRotation * 3.14f / (float)180) *
20.0;
    SSD1306_DrawLine(screen_width/2, screen_height/2, screen_width/2 +
comppasLineX, screen_height/2 + comppasLineY, 1);
    SSD1306_UpdateScreen(); // update screen
}
//Busiusti kampo informacija i kompiuteri per UART
void Update_PC_Heading()
{
    sendData[0] = 0x03; //Komandos pavadinimas
    sendData[1] = compassRotation >> 8;
    sendData[2] = compassRotation & 0xFF;
    HAL_UART_Transmit_IT(&huart4, sendData, 3);
}
//NUsiusti i kompiuteri esama kalibracijos reiksme
void Update_PC_Calibration()
{
    sendData[0] = 0x04; //Komandos pavadinimas
    sendData[1] = calibrationValue >> 8;
    sendData[2] = calibrationValue & 0xFF;
    if(HAL_UART_Transmit(&huart4, sendData, 3, 1000)  != HAL_OK){

    }
}
//Perskaityti kalibracijos reiksme
void Read_Calibration()
{
    uint32_t dataIn[1];
    Flash_Read_Data(0x0803F800, dataIn, 1); //Bibliotekos funkcija
    calibrationValue = (dataIn[0] >> 16);
}
//Irasyti i atminti nauja kalibracijos reiksme
void Update_Calibration(uint16_t new_calibration)
{
        uint32_t data[1];
        data[0] = new_calibration << 16;
        Flash_Write_Data(0x0803F800, data, 1); //Bibliotekos funkcija
}
```

2 Kompiuterio programos kodas

Dizaino langas:




```
 private bool ConnectController(string selectedPort)
{
    if (!isConnected)
    {
        try
        {
            port = new SerialPort("COM3", 9600, Parity.None, 8, StopBits.One);
            port.Open();
            port.DataReceived += new
SerialDataReceivedEventHandler(Port_DataReceived);
            isConnected = true;
            connectedPort = selectedPort;
            ChangeDisplayedInfo("Connected to port " + selectedPort);
            return true;
        }
        catch
        {
            isConnected = false;
            ChangeDisplayedInfo("Failed to connect");
            return false;
        }
    }
    return false;
}
```

```
private void Port_DataReceived(object sender, SerialDataReceivedEventArgs e)
//Receive USB data
{

    int count = port.BytesToRead;
    byte[] ByteArray = new byte[count];
    port.Read(ByteArray, 0, count);
    Debug.WriteLine("Got" + ByteArray.Length);

    int skaitmuo = 0;

    if (ByteArray.Length > 0) {
        for (int i = 0; i < ByteArray.Length; i+= 3)
        {
            switch ((int)ByteArray[i])
            {
                case (3):
                    skaitmuo = ByteArray[i+1] << 8 | ByteArray[i+2];
                    Update_Compass(skaitmuo);
                    Debug.WriteLine("Got: " + skaitmuo);

                    string timeData = DateTime.Now.ToString("h:mm:ss");
                    compassData.Add(new CompassData(skaitmuo, timeData));

                    break;
                case (4): //Gavome kalibracija
                    skaitmuo = ByteArray[i+1] << 8 | ByteArray[i+2];
                    Update_Claibration(skaitmuo);

                    break;
            }
        }

    }

}
```

```
private void groupBox1_Paint(object sender, PaintEventArgs e)
{
    int comppasLineX = -(int)Math.Round(Math.Sin(((float)compass_angle+
(float)180)*Math.PI / (float)180) * 50);
    int comppasLineY = (int)Math.Round(Math.Cos(((float)compass_angle +
(float)180) * Math.PI / (float)180) * 50);

    Color black = Color.FromArgb(255, 0, 0, 0);
    Pen blackPen = new Pen(black);
    blackPen.Width = 2;
    e.Graphics.DrawLine(blackPen, CompassBox.Width / 2, CompassBox.Height / 2,
CompassBox.Width/2 + comppasLineX, CompassBox.Height / 2 + comppasLineY);

    Color green = Color.FromArgb(255, 0, 255, 0);
    Pen greenPen = new Pen(green);
    greenPen.Width = 5;
    e.Graphics.DrawEllipse(greenPen, CompassBox.Width / 2-25,
CompassBox.Height / 2-25, 50, 50);
}



private void ConfigButton_Click(object sender, EventArgs e)
{
    if(ConfigTextBox.Text.Length > 0)
    {
        try
        {
            int textValue = Int32.Parse(ConfigTextBox.Text);
            byte[] intBytes = BitConverter.GetBytes(100);
            byte[] sendByte = new byte[3];
            sendByte[0] = 0x01;
            sendByte[1] = (byte)((textValue >> 0) & 0xFF);
            sendByte[2] = (byte)((textValue >> 8) & 0xFF);

            port.Write(sendByte, 0, 3);
        }
        catch (Exception ee)
        {
            ConfigTextBox.Text = "";
        }
    }
}


public struct CompassData
{
public int compassHeading;
public string time;

public CompassData(int _compassHeading, string _time)
{
    compassHeading = _compassHeading;
    time = _time;
}
}

private void SaveToFileButton_Click(object sender, EventArgs e)
{
    string fullDateTime = DateTime.Now.ToString("yyyy_MM_dd_h_mm_ss");
    TextWriter txt = new
StreamWriter(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, (fullDateTime
+ "_compass_data.txt")));
    txt.Write("Heading | Time \n");
    foreach (CompassData c in compassData)
    {
        txt.Write(c.compassHeading.ToString() + " | " + c.time + "\n");
    }
    txt.Close();
}
```