eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Gradu Amaierako Lana

Konputagailuen Ingeniaritza

Minimeteo: Atmega328p mikrokontrolagailuan oinarritutako estazio meteorologikoa

2022

Unai Fernandez Urroz

Zuzendariak

Jose Ignacio Martin eta Izaskun Etxeberria

Laburpena

Proiektu honetan, uneko eguraldiari buruzko hainbat parametro meteorologikoen egoera emango duen estazio meteorologikoa eraiki da. Egiten diren neurketa meteorologikoen artean, tenperatura, hezetasuna eta haizearen abiadura aurki ditzakegu. Tenperatura eta hezetasuna neurtzeko DHT-11 sentsorea erabili da eta haizearen abiadura, anemometro baten bidez lortu da.

Minimiteo estazioak, eguraldiaren inguruko datuak eskuratzeaz gain, lortutako informazioa WiFi bidez bidaltzeko ahalmena edukiko du. WiFi transmisioetarako erabili den ESP-8266 modulua, Access Point moduan egongo da konfiguratuta, erabiltzaileek datuak eskura ditzaten.

Estazioan erabiltzen diren sentsore eta modulu desberdinen kontrolerako, Atmega328p mikrokontroladorea erabili da. Txip hau oso ezaguna da Arduino UNO txartelak erabiltzen duelako eta atzean daukan komunitate handiak hamaika liburutegi ditu eskuragarri, era askotako proiektuen garapenerako. Dena den, proiektu honetan liburutegi eta baliabide guzti horiek alde batera utzi dira eta mikrokontroladorea erregistro-mailan programatu da, hau da, behe-mailan.

Gaien Aurkibidea

Irudien Zerrenda

1. KAPITULUA

Sarrera

Kapitulu honetan, garatutako proiektuari sarrera bat egingo zaio. Lehenengo, burututako lanari testuinguru bat emanez eta proiektua gauzatzeko egon den motibazioa azalduz. Ondoren eraikitako estazioaren deskribapena egingo da, honen egitura argi edukitzeko eta azkenik dokumentu honetan landuko diren kapitulu desberdinen laburpena egingo da.

1.1 Testuingurua

Meteorologia atmosferan gertatzen diren fenomeno fisikoak aztertzen dituen zientzia da. Meteorologiaren kontzeptua momentu eta leku zehatz batean atmosferak daukan egoerarekin erlazionatzen da. Uneko tenperaturak, prezipitazioek, haizeak edo beste faktore desberdinek determinatuko dute egoera hori.

Historian zehar aldaketa meteorologikoak eta hauek inguruko kliman daukaten eragina aztertzea oso garrantzitsua izan da. Esaterako, nekazaritzan ezinbestekoa da datu hauen ezagutza, uztak ahalik eta emankorragoak izan daitezen. Horretaz gain, nabigazioan eta operazio militarretan ingurunearen egoera meteorologikoa ezagutzea funtsezko faktore bat da.

Aldagai meteorologiko guzti hauek modu erregular batean neurtzeaz, estazio meteorologikoak arduratuko dira. Estazio hauek jasotako datuak, eredu matematikoak erabiliz, eguraldiari buruzko iragarpenak egiteko erabiliko dira. Munduko meteorologia erakundearen arabera, hainbat estazio mota desberdindu ditzakegu. Hona hemen ezagunenak:

- **Estazio meteorologiko sinoptikoa:** Eguraldiaren behaketa denbora tarte motzetan egiten du, eta lurrazalean daude kokatuta. Zeruaren egoera, hodeien altuera, presio atmosferikoa itsas mailan, tenperatura, haizearen norabidea, eta abiadura eta prezipitazioa neurtu ohi ditu.
- **Aireko estazio meteorologikoa:** Atmosferako goiko geruzak aztertzeko balio dute. Hainbat metodo desberdin erabiltzen dira hau egiteko, horien artean globoak aurki ditzakegu. Globoa airera bidaltzen da eta haizearen abiadura eta norabidea, tenperatura eta altitudeari buruzko datuak lortzen ditu altuera desberdinetan.
- **Nekazaritzarako estazio meteorologikoa:** Izenak dioen bezala nekazaritzan ahalik eta etekin handiena ateratzeko erabiltzen dira ingurua aztertzeko. Lurrazalean daude kokatuta eta lehen aipatutako neurketetaz aparte, lurraren hezetasuna bezalako parametroak neurtzeko gai dira.
- **Estazio meteorologiko automatikoa:** Izenak esaten duen moduan, ohiko estazio baten bertsio automatikoa da. Behaketa eta aldagai bakoitzaren eguneraketa guztiak era automatikoan egiten ditu. Orokorrean lurrazaleko fenomeno meteorologikoak aztertzeko balioko dute.

1.2 Deskribapena

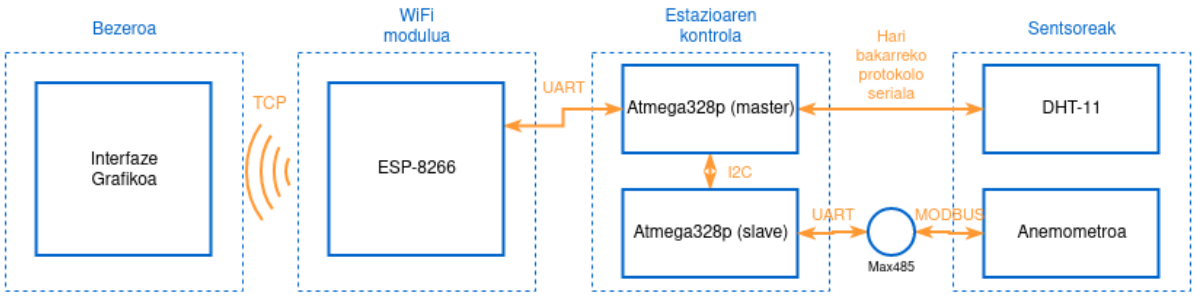
Proiektu honetan, estazio meteorologiko automatiko bat eraikiko da. Estazioak sentsore desberdinen datuak jaso eta Wifi bidez bidaliko ditu, erabiltzaileak ordenagailuan aplikazio baten bidez ikus ditzan.

Proiektuak, 1.1 irudian ikusi daitekeen bezala, lau atal nagusi ditu: 1) datuak jasozteko sentsoreak; 2) estazioaren kontrolaz arduratuko diren bi *Atmega328p* Mikrokontroladore; 3) wifi modulua eta 4) bezeroaren interfaze grafikoa. Tenperatura, hezetasuna eta haizearen abiadura lortzeko bi sentsore desberdin erabili dira: *DHT-11* eta anemometro bat. Mikrokontroladoreak sentsoreekin komunikatzeko hainbat protokolo erabili dira. *DHT-11*-ren kasuan, norabide

bikoitzeko hari bakarreko serie protokolo espezifikoa erabili behar da. Anemometroaren kasuan, berriz, *Max485* modulu baten bidez, anemometroak erabiltzen duen *MODBUS* protokoloa *UART* protokolora bihurtuko da, *Atmega328p* txiparekin komunikatu ahal izateko. 1.1 Irudian ikus daitekeen bezala bi mikrokontroladore daude, bata nagusia eta bestea morroia. Bi mikrokontroladore erabiltzearen arrazoia, txip hauek *UART* modulu bakarra daukatela da. Hori dela eta Atmega328p batek ezingo lukeen *ESP-8266* Wifi modulua eta anemometroarekin aldi berean komunikatu. Mikro nagusiak morroiarekin komunikatzeko anemometroaren datuak eskura ditzan, *I2C* protokoloa erabiliko da.

ESP-8266 Wifi modulua, lehen aipatu bezala, *UART* bidez komunikatuko da estazioko mikrokontroladore nagusiarekin. Bestalde, Wifi modulua Access Point eta zerbitzari moduan konfiguratuko da, bezeroak estazioaren Wifira zuzenean konektatu ahal izateko.

Azkenik, estazioko sentsoreek lortutako datuak *python*-en garatutako aplikazio baten bidez bistaratuko dira. Aplikazioak *TCP* protokoloaren bidez komunikatuko da Wifi moduluarekin, horrela datuen osotasuna, zuzentasuna eta ordena mantentzea bermatzen delako.



1.1. Irudia: Estazioaren egitura.

1.3 Dokumentuaren kapituluak

Dokumentu honetan, proiektua aurrera eramateko erabilitako material eta kontzeptuen azalpena aurkituko da, hainbat kapitulutan banatuta.

- **Proiektuaren Helburuen dokumentua:** Kapitulu honetan, proiekturako zehaztu diren helburuen, proiektua garatzeko egindako plangintza, denbora estimazioen eta arriskuen azalpena emango da.
- **Mikrokontroladorea:** Proiekturako erabili den mikrokontroladorearen azalpena emango da eta estazioa funtzionarazteko erabili diren barne modulu guztiei buruzko argibideak emango dira.
- **Erabilitako sentsore eta moduluak:** Kapitulu honetan, estazioak egin beharreko neurketak egiteko erabili diren sentsore eta moduluen azalpena emango da.
- **Erabiltzailearen interfazea:** Estazioak irakurritako datuak bistaratzeko egindako aplikazioaren ezaugarrien eta erabilitako teknologiaren azalpena emango da.
- **Sistemaren garapena:** Sistema osoa osatzen duen kodearen egitura orokorra aztertuko da.

- **Ondorioak:** Azkeneko kapituluan proiektuaren ondorioak eta etorkizunean gehitu nahi diren hobekuntzen aurkezpena egingo da.

2. KAPITULUA

Proiektuaren Helburuen Dokumentua

Kapitulu honetan, proiekturako zehaztu diren helburuak aurkeztuko dira, eta hau garatzeko egin diren plangintza, denbora estimazio eta arriskuen azalpena emango da.

2.1 Helburuak

Proiektu honen helburu nagusia, Arduinok erabiltzen duen *Atmega328p* mikrokontroladorea erregistro mailan programatuta estazio meteorologiko bat eraikitzea da. Estazioak sentsore desberdinen informazioa jasotzeko gai izango da eta Wifi bidez beste gailu batera transmitituko ditu.

Proiektu honetan erregistro mailan lan egiteak zailtasun maila igoko du, mikrokontroladorearen *barne-moduluen* funtzionamendua zuzena izateko, erregistroak behar bezala konfiguratzea lan nekeza baita. Arduinoren inguruneak hamaika liburutegi eta tresna eskaintzen ditu, sentsore eta modulu desberdinak kontrolatzeko, eta modu horretan proiektuen garapena asko errazten da. Kasu askotan ordea, zaila da liburutegi horien atzean dagoena ezagutzea. Proiektuko kodea erregistro mailan garatzeak hainbat abantaila izango ditu, horien artean, *Atmega328p* mikrokotroladorearen barne egitura xehetasun gehiagorekin ezagutzea esaterako. Garapena modu honetan eginda, *Sistema Txertatuen Diseinua* irakasgaian landutako hainbat kontzeptu *Arduino* bezalako ingurune batera moldatu beharko dira.

Estazio meteorologikoa osatuko duten *ESP-8266* Wifi modulua eta eguraldiari buruzko datuak lortzeko balioko duten sentsoreen azterketa sakona egin behar da, horren integrazioa proiektuan zuzena izan dadin.

Azkenik, sentsore desberdinetatik lortutako datuak ikuskatzeko aplikazio bat eskuz diseinatuko da. Lehen aipatu bezala sentsoreek jasotako datuak Wifi bidez beste gailu batera bidaltzea da ideia, eta beste gailutik aplikazio baten bidez beharrezko informazioa eskuratzea. Aplikazioa eskuz garatuko da *Sare Zerbitzu eta Teknologiak* irakasgaian landutako TCP bezeroa oinarri moduan hartuta, ordenagailuan datuak ikus daitezen.

2.2 Lan paketeak eta atazen deskonposaketa

Atal honetan 2.1 Irudian ikusi daitezkeen LDE diagramako lan pakete eta ataza desberdinen deskonposaketa egingo da eta bakoitzaren azalpen laburra emango da.

Kudeaketa:

- **Plangintza (P)** lan-paketeak, hasierako plangintza egiteko atazak eta behar badira, plangintza egunean mantentzeko behar daitezkeenak barne izango ditu.

  - **P.1:** Eskakizunen identifikazioa, hasierako erabakiak hartzea, informazioaren analisia eta zalantzen ebazpena.

  - **P.2:** Hasierako plangintza, garapenerako ingurunea prestatzeko.

  - **P.3:** Plangintzaren eguneraketa, beharrezkoa bada.

- **Jarraipen eta kontrola (JK)** lan-paketeak proiektuaren garapen egokia bermatuko duten atazak edukiko ditu eta konkretuki, proiektuko ataza bakoitzerako dedikazioen jarraipena eta epeen eta espezifikazioen betetzea.

  - **JK.1:** Proiektuaren garapenari buruzko informazio garrantzitsua jaso.

- **JK.2:** Planarekin, jarraipeneko informazioaren konparaketa, desbideratze esanguratsuen eta sortzen diren arriskuen identifikazioa.

- **JK.3:** Proiektuak arrakasta izan dezan, baldintzen bermatzea.

Garapena:

- **Ingurunearen prestaketa (IP)** lan paketean, proiektuan zehar estazioaren sorketa aurrera eramateko erabiliko diren baliabide guztien identifikazioa eta ikerkuntza sartzen dira.

  - **IP.1:** Beharrezkoa den softwarea identifikatu eta deskargatu.

  - **IP.2:** Baliabideen azterketa.

- **Wifi modulua (W)** lan-paketeak, **ESP8266** Wifi moduluarekin zerikusia daukaten ataza guztiak edukiko ditu.

  - **W.1:** Wifi moduluaren ikerketa.

  - **W.2:** Wifi modulua eta mikrokontroladorea elkarlanean egoteko liburutegia sortu.

  - **W.3:** Sentsoreetatik jasotako datuak bidaltzeko protokolo baten diseinua.

- **Sentsoreak (S)** lan-paketeak, eguraldia aztertzeko balioko duten sentsoreen ikerketarekin zerikusia dauzkaten zereginak elkartuko ditu.

  - **S.1:** Erabili beharreko sentsoreen ikerketa.

  - **S.2:** Sentsoreak eskuratu edo eraiki.

  - **S.3:** Sentsoreak kontrolatzeko kodea.

- **Programa Nagusia (PN)** lan-paketeak, aurreko paketeetan garatutako kodea programa nagusi batean biltzea du helburu.

  - **PN.1:** Wifi moduluaren kodea programa nagusian sartu.

  - **PN.2:** DHT-11 sentsorearen kodea programa nagusira gehitu.

  - **PN.3:** Anemometroaren kodea programa nagusian sartu.

- **Aplikazioaren diseinua eta garapena (AP)** lan-paketea, datuak bistaratzeko aplikazio bat diseinatzeko eta garatzeko atazek osatuko dute.

  - **E.1:** Aplikazioaren egitura diseinatu.

  - **E.4:** Aplikazioaren garapena.

Dokumentazioa:

- **Memoria (M)** lan-paketeak proiektuari buruzko memoria bat idaztea dauka helburu.

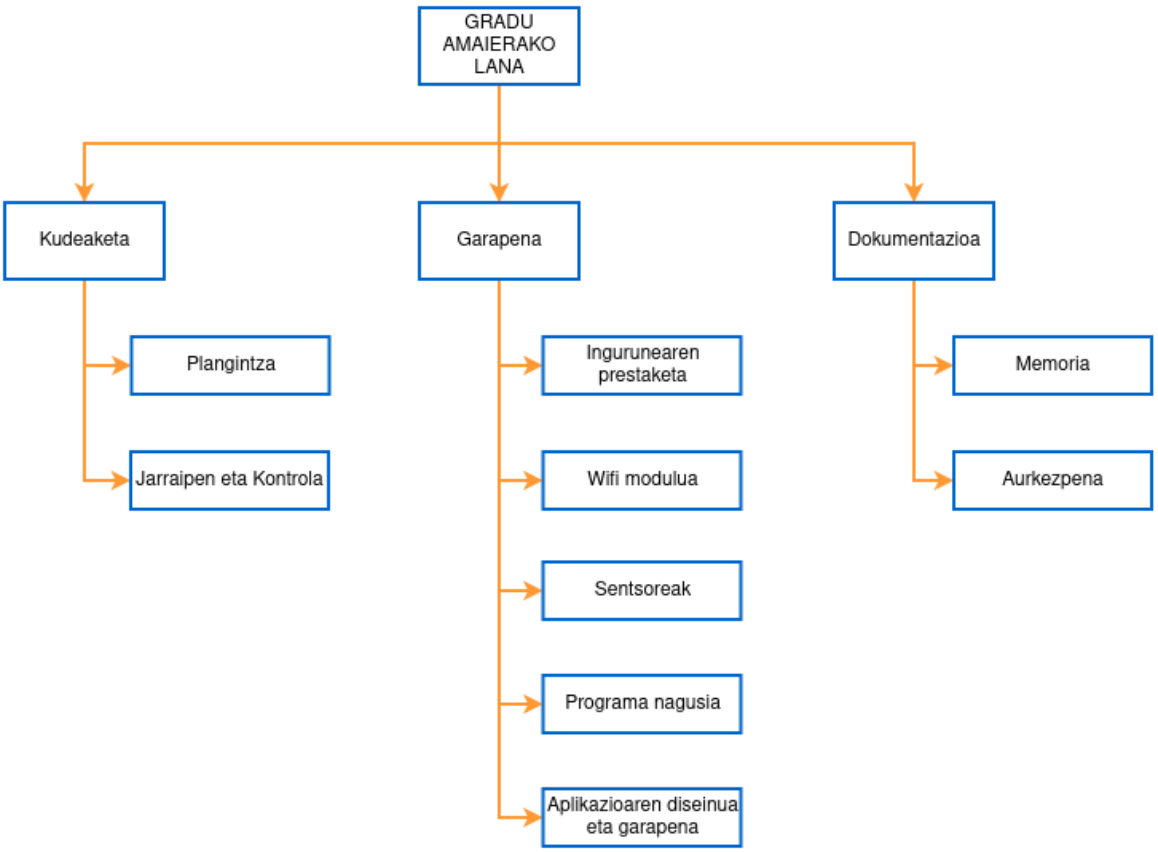
  - **M.1:** Memoriaren egitura prestatu.

  - **M.2:** Memoria osatu, proiektua nola garatu den eta plangintza azalduz.

- **Aurkezpena (A)** lan-paketearen barruan defentsa egunerako beharrezkoa izango den aurkezpena prestatzeko zereginak dauzka.

  - **A.1:** Aurkezpenaren garapena.

  - **A.2:** Aurkezpenaren prestaketa.



2.1. Irudia: LDE diagrama

2.3 Denbora estimazioak eta desbiderapenak

2.2 Irudiko taulan, aurreko ataleko LDE diagraman definitutako lan paketeen denbora estimazioak agertzen dira.

Ikusten den bezala garrantzi handiena edukiko duten lan paketeak *Garapena* eta *Dokumentazioa* dira. Nahiz eta *Kudeaketan* ez diren hainbeste ordu eskainiko ezinbestekoak dira bertako

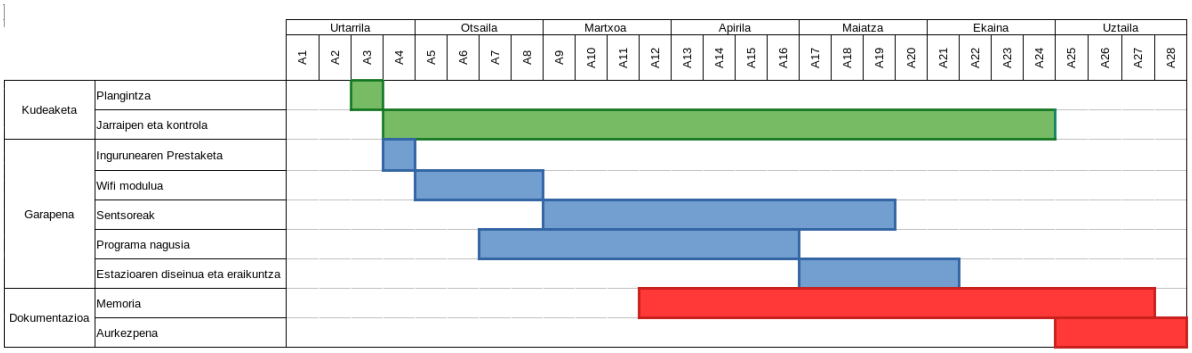
	Lan paketea	Ordu estimazioa	Ordu kopurua
Kudeaketa	Plangintza	8	8
	Jarraipen eta kontrola	15	15
Garapena	Ingurunearen prestaketa	7	7
	Wifi modulua	55	42
	Sentsoreak	50	60
	Programa Nagusia	55	65
	Aplikazioaren garapena	25	30
Dokumentazioa	Memoria	75	75
	Aurkezpena	15	15
Guztira		305	317

2.2. Irudia: Lan paketeen ordu kopurua

atazak zuzen burutzea proiektuaren garapena espero den bezala joateko. Taulan ikusten den bezala guztira 305 ordu estimatu ziren proiekturako, baina azkenean gehiago izan dira proeiktuari dedikatu zaizkion ordu kopurua. Orokorrean ez da desbiderapen asko egon ordu kopuru orokorra kontuan hartuta, azkenean 317 ordu dedikatu baitzaizkio proiektuari.

Desbiderapen nabarienak garapenean aurkituko ditugu. Wifi moduluari estimatutakoak baino 8 ordu gutxiago eskaini zaio, Sentsoreekin berriz, hasieran pentsatutakoa baino 10 ordu gehiago eman zaizkio. Azkenik, programa nagusiaren garapenari ere 10 ordu gehiago eskaini zaizkio.

2.3 Irudiko gantt diagramaren bidez, lan paketa hauen garapena denboran zehar ikus dezakegu. Era honetan errazagoa izango da ataza bakoitzaren hasiera eta bukaera datak egutegian finkatzea.



2.3. Irudia: GANTT diagrama

2.4 Kalitate plana

Proiektuaren kalitate maila finkatzeko proiektuak bete beharko duen oinarrizko kalitate maila zehaztu behar da. Atal honetan proiektuak kalitate ona izan dezan, behar dituen ezaugarriak aipatuko dira.

Oinarrizko betekizunak:

- Wifi bidezko komunikazioa egon behar du estazioaren eta beste gailu baten artean.
- Sentsoreen funtzionamendua zuzena izan behar du.
- Proiektuko Hardware guztia babesteko egitura bat eraiki.
- Sentsoreek jasotako datuak beste gailu batetik kontsultatzeko ahalmena
- Kodea txukuna eta ulergarria izan behar du, interneten eskuragarri egongo baita edonorentzat.

2.5 Arriskuak

Proiektuaren garapena ondo joan dadin ezinbestekoa da egon daitezkeen arazoak identifikatzea, hauei soluzio bat bilatzeko. Arrisku guztiek ez dute eragin berdina izango proiektuarengan, eta hauek gertatzeko probabilitatea ez da berdina izango. Honako hauek dira proiektuan zehar gerta daitezkeen arazo batzuk.

- **Hardwarearekin arazoak:** Mikrokontroladoreetarako softwarea garatzerako orduan ohikoa da arazoak edukitzea, batzutan plakaren eta sistema eragilearen arteko komunikazioa ez dabil ondo edo sentsoreek emandako emaitzak ez dira esperotakoak.

  - **Probabilitatea:** Altua

  - **Soluzioa:** Arazoari buruz informatu eta lehenbailehen zuzentzen ahalegindu. Erabilitako iturriak gordetzea komenigarria da berriro gertatu ezkero eskura izateko.

- **Datuen galera:** Proiektuan aurreratutako lan guztia galtzeko aukera badago. Hau gertatzeak, galdu den informazioaren arabera arazo handia izan liteke beraz ezinbestekoa da datuak ondo gordetzea.

  - **Probabilitatea:** Ertaina.

  - **Soluzioa:** Informazio sistema ondo antolatua, eta segurtasun kopiak eduki. Komenigarria da *Google Drive* bezalako plataforma batean datuak gordetzea orokorrean arazorik ematen ez duelako.

- **Denbora falta:** Baliteke aurreko arazoen ondorioz edo planifikazio txarra egiteagatik proiektua bukatzeko denbora falta izatea, horrek entrega atzeratzera behartuko luke.

  - **Probabilitatea:** Ertaina.

  - **Soluzioa:** Planifikazioa kontu handiz egin ahalik eta xehetasun gehienak kontua hartuz.

3. KAPITULUA

Mikrokontroladorea

Kapitulu honetan, proiektua aurrera eramateko erabili den mikrokontroladorea aztertuko da. Gailu honek dituen barne moduluak ezagutzera emango dira ere, hauen funtzionamendua azalduz eta estazioa behar den bezala funtzionatzeko konfiguratu behar diren erregistroak komentatuz.

3.1 Atmega328p mikrokontroladorea

Proiektu honetarako erabili den mikrokontroladorea *Atmega328p* izan da. Gailu hau *Atmel* konpainiak garatutako *megaAVR* serieko potentzia baxuko 8 biteko CMOS mikrokontroladore bat da, AVR-rako hobetutako RISC arkitekturan oinarritutakoa. Mikrokontroladore hau oso ezaguna da, Arduino UNO txartelak erabiltzen baitu.

Arduino UNO plaka oso erabilia da elektronikako proiektuak garatzen hasiberriak direnen artean, merkea eta programatzeko erraza baita. Arduino plataforma handia bihurtu da 2003 urtean Arduino UNO-a atera zutenetik, eta gaur egun hamaika liburutegi eta plaka berri atera dituzte komunitatearen nahiak asetzeko, eta horrek erraztasun handiak ematen ditu programatzerako orduan. Dena den, plataformak eskaintzen dituen baiabide horiek guztiak alde batera utziko dira, programatzerako orduan erabiltzen diren funtzio eta liburutegi guztiak eskuz garatuko baitira, hau da, erregistro-mailan. Honi esker erabiltzen diren barne moduluen behe-mailako konfigurazioa xehetasun guztiekin aztertu ahal izateko aukera dago.

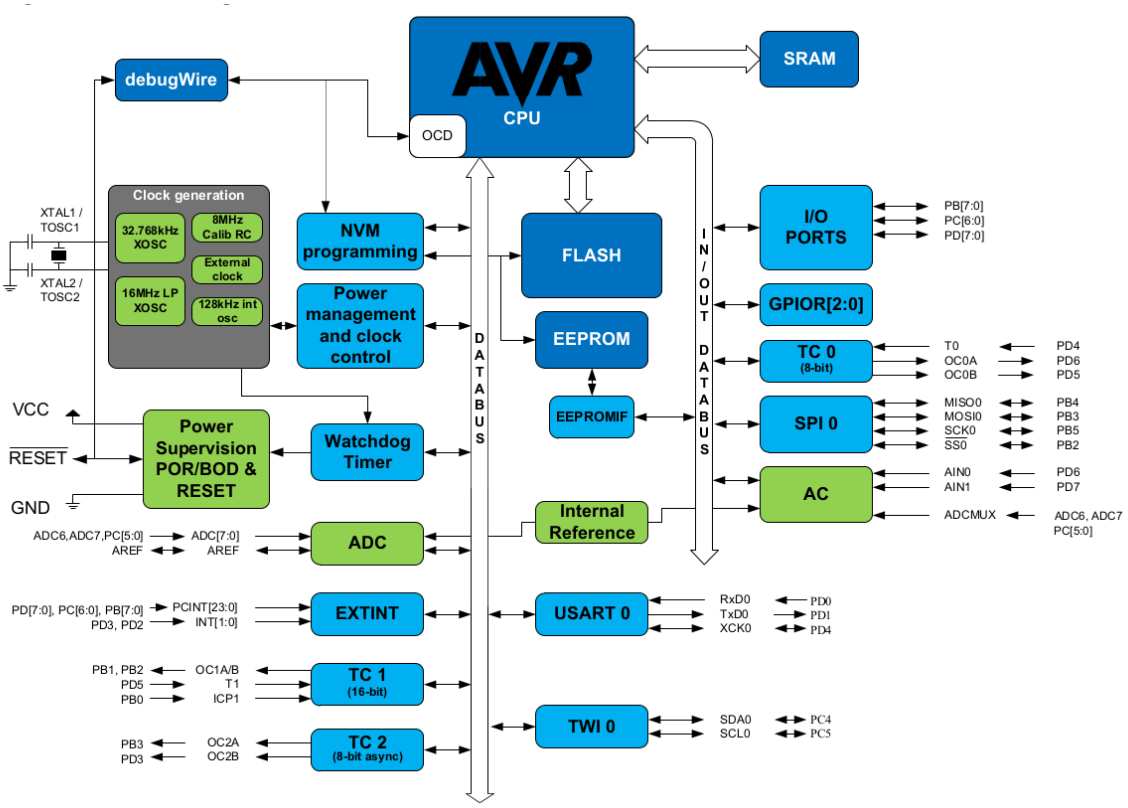
Aipatu bezala, Atmega328p txiparen erregistroak eskuz programatuta barne moduluen xehetasunak ezagutzeko ahalbidetuko gaitu, baina programatzeko zailtasuna igoko da, beharrezkoak diren erregistroen bit bakoitza arretaz kontrolatu behar baita. Erregistroen funtzionamenduarekin sartu aurretik, mikrokontroladorearen ezaugarrien azterketa bat egingo da.

3.1.1 Barne egitura

Atal honetan, Atmega328p mikrokontroladoreak daukan barne egitura aztertuko dugu. 3.1 irudian txiparen bloke diagrama ikus dezakegu. Bertan, barne modulu guztiak agertzen dira, txiparen hankatxoekin edota beste barne modulu batekin dauzkaten konexioak irudikatuz. Honi esker Arduinoak erabiltzen duen mikrokontroladorearen egituraren mapa orokor bat osatu dezakegu. Jarraian irudi horretan agertzen diren osagai nagusien azalpena emango da.

- **AVR prozesu unitatea:** Hau da mikrokontroladorearen zati garrantzitsuena, programen exekuzio zuzena bermatuko baitu. Prozesu unitateak *Harvard* arkitektura erabiltzen du, programa eta datuetarako bus eta memoria desberdinak erabiliz. Agindu bakoitza erloju ziklo batean exekutatzen da eta memoriak atzitzeko, kalkuluak egiteko, periferikoak kontrolatzeko eta etenak kudeatzeko gai da.
- **Memoriak:** Hiru dira Atmega328p-ak dituen memoriak, 32 kbyteko Flash memoria programak gordetzeko, 2 kbyteko SRAM memoria bat, programak sortzen dituen aldagaiak gorde ahal izateko eta azkenik 1 kbyteko EEPROM memoria bat informazio iraungarria gorde ahal izateko.
- **ADC modulua:** seinale analogiko bat 10 biteko datu batean bihurtzen du(0-tik 1023-ra). Modulu honek 6 sarrera hankatxo ditu.
- **Komunikazio interfazeak:** Hiru desberdin dauzka txip honek, USART, SPI eta TWI (I2C). Interfaze horietako bakoitzak datuak jaso eta bidali ahal izateko aurredefinitutako eta komunikaziorako prest dauden sarrera/irteera hankatxoak erabiltzen dituzte.
- **Tenporizadoreak:** Guztira 3 tenporizadore daude (TC0, TC1 eta TC2), horietako bi 8 bitekoak izanik eta bestea 16 bitekoa.

- **Etenen kontroladorea:** Eten baten bidez, une zehatz batean exekutatzen ari den kodea eten egingo da, zerbitzu-errutina bat exekutatzeko. Bukatzerakoan, programa nagusiak bere exekuzioa jarraituko du. Etenen kontroladorea eten horiek kudeatzeaz arduratuko da.
- **Erloju sistema:** Mikrokontroladore batek egin beharreko atazak eta konektatutako periferikoak sinkronizatzeko erloju seinale bat sortzen du. Proiektuan erabilitako Atmega328p mikrokontroladorearen kasuan 16MHz koa izango da erloju seinalearen maiztasuna.



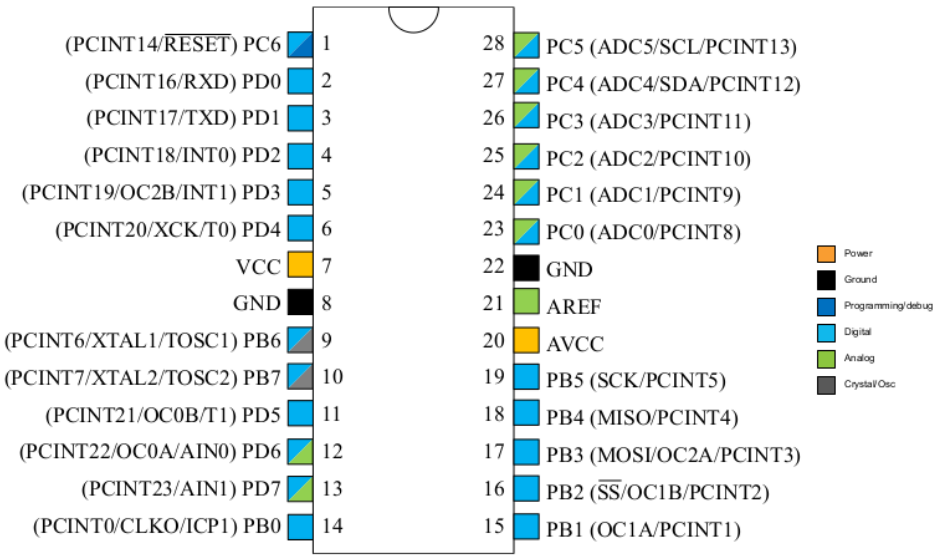
3.1. Irudia: Atmega328p txiparen bloke diagrama ([atm, 2016])

3.1.2 Sarrera/Irteera hankatxoak

Atal honetan *Atmega328p* txipak dituen hankatxoak aztertuko ditugu. 3.2 irudian ikus daitekeen bezala, 28 hankatxoz osatuta dago mikrokontroladorea. *AVR* txip batean, ia hankatxo guztiak sarrera edo irteera moduan konfiguratu daitezke. Hau da hankatxo bakoitza, barnealdean, transistore edo tentsio aldaketei sentikorra den elementu bati dago konektatuta. Portaera hau eskuz zehaztu ahal izateko erregistroa batzuk erabiltzen dira. Erregistro horiek memorian mapeatuta daude, aldagai arrunt bat bezala funtziona dezaten eta, horretaz gain, fisikoki daude lotuta barne zirkuituari.

Atmega328p txipean, 3.2 irudian ikusi daitekeen bezala, hiru hankatxo multzo desberdinetan banatzen dira: *PORTB, PORTC eta PORTD*. Hankatxo talde hauetako bakoitzak, 3 erregistro garrantzitsu ditu *hankatxoak* konfiguratu ahal izateko.

- **Data-direction register (DDR)**: Erregistro honen bidez, hankatxoa sarrera edo irteera moduan konfiguratuko da. Erregistroko bitaren balioa 0 bada, sarrera moduan egongo da konfiguratuta eta 1 bada irteera moduan. Defektuz, sistema pizten denean 0 balioa daukate erregistro guztiek (denak sarrera gisa).
- **Port data register (PORT)**: Erregistro honek funtzionatzeko bi modu ditu DDR erregistroak hartzen duen balioaren arabera. DDR-ren balioa 1 bada (output), hankatxoaren balio logikoa esleitzeko balioko du. Aldiz, DDR-ren balioa 0 bada (input), hankatxora barne erresistentzia aktibatzeko balioko du.
- **Port input pins address (PIN)**: Hankatxo bakoitzaren egoera irakurtzeko balio du.



3.2. Irudia: Atmega328p txiparen hankatxoen diagrama

3.1.3 Etenen kudeaketa

Sistema txertatu batean etenak baliabide oso erabilgarriak dira. Mikrokontroladorera eten eskaera bat iristen denean, une zehatz horretan exekutatzen ari den programa gelditu egiten da, eten zerbitzu errutina bat exekutatzeko (ISR). Zerbitzu errutina bukatzerakoan lehen gelditutako programa berriz martxan jartzen da.

Atmega328p txipak 25 iturri desberdinetatik jaso ditzake eten eskaerak. Iturri hauetako bakoitza txipeko Flash memoriako helbide txikienetan kokatuta dauden etenei daude lotuta. Helbide hauek eten bektoreak dira eta memorian daukaten posizioaren arabera lehentasun handiagoa edo txikiagoa edukiko dute. Hau da, memoriako helbide txikienek izango dute lehentasun handiena. Atmegaren kasuan *RESET* izango da lehentasun handiena daukan etena.

Eten bakoitza gaitzeko eta desgaitzeko bit bat daukate lotuta. Bit horren balioa batekoa izan behar du etena erabili ahal izateko.

Eten zerbitzu errutina batean exekutatuko den kodea zehazteko *ISR* funtzioa erabiliko da eta parametro moduan zehaztuko da erabiliko den eten bektorea izenaren bidez.

```
ISR(TIMER0_COMPA_vect){

}
```

Aurreko adibidean T0 tenporizadorearen zerbitzu errutina ikus daiteke. Aipatu bezala erabiliko den eten bekorea parentesien artean zehaztuko da eta funtzioaren barruan idatzitako kodea eten bat sortzen den bakoitzean exekutatuko da.

3.2 USART

USART, Universal Syncronous Asynchronous Receiver-Transmitter sigletatik dator. Komunikaziorako serie protokolo bat da, hau da, datuak bitez bit bidaliko dira, pisu txikieneko bitarekin hasiz. Protokolo honen abantaila nagusia komunikazioa burutzeko kable gutxi behar direla da, bata transmititzeko (TX) eta bestea jasotzeko(RX). Bi noranzkoko komunikazioa eduki arren, ezin dira biak aldi berean erabili.

Atmega328p-ren kasuan serie komunikazioaren kontrolaz arduratuko den *USART* modulu bakarra dago eta 2 eta 3 hankatxoetara dago konektatuta (ikus 3.2 irudia). Proiektu honetan, *USART* protokoloa bi kasu desberdinetan erabili da. Lehenengoan, ESP-8266 Wifi txartelarekin komunikatzeko eta bigarrenean, anemometroaren datuak eskuratzeko. Lehen aipatu bezala, estazioa eraikitzeko erabilitako mikrokontroladoreak *USART* modulu bat baino ez dauka, beraz, bi gailuekin komunikatu ahal izateko bi mikrokontroladore erabili dira.

Izenak dioen moduan, *USART* moduluak modu sinkrono zein asinkronoan funtzionatzeko ahalmena dauka. Modu sinkronoan datuak bidaltzeaz gain, komunikazioan parte hartzen duten bi muturren artean sinkronizazio seinale bat dago. Modu asinkronoan berriz, ez da sinkronizaziorako seinalerik erabiltzen, eta igorleak kontrol bit bereziak bidaltzen ditu: Start bita, Stop bita eta paritate bita. Horretaz gain, datuak ondo jasotzeko, bai igorleak eta hartzaileak transmisio abiadura berdina izan beharko dute.

Nahiz eta Atmega328p mikrokontroladoreak *USART* modulua izan, modu asinkronoan egingo da lan, erabilitako ESP-8266 Wifi txartelak eta Anemometroak era horretan lan egitera behartzen baitute.

3.2.1 Moduluaren konfigurazioa

Atal honetan ESP-8266 Wifi moduluarekin eta anemometroarekin komunikatzeko *USART* moduluaren konfigurazioa azalduko da. 3.3 Irudian, igorle (Atmega328p) eta hartzaileen (wifi modulua) artean egongo den oinarrizko komunikazioaren irudikapen grafikoa ikusi daiteke. Komunikazio bakoitzaren hasiera start bitak finkatuko du igorleak, ondoren 8 biteko datu bat bidaliko da, eta mezuaren bukaera stop bitaren bitartez zehaztuko da.

3.3. Irudia: UART komunikazio baten irudikapen grafikoa

USART modulua behar den bezala konfiguratzeko Atmega328p mikrokontroladoreko erregistro batzuen biten balioak aldatu behar dira. Jarraian, *USART* moduluaren hasieraketak egiteko erabilitako erregistroetako biten azalpena emango da.

UCSR0C

Jarraian, 3.4 Irudian ikus daitekeen UCSR0C erregistroko bitei balio desberdinak emanda konfiguratutako USART moduluaren funtzioen azalpena emango da.

1. USART modulua asinkrono moduan konfiguratu

   Lehen aipatu den bezala, Atmega328p txipak daukan USART moduluak modu sinkrono zein asinkronoan lan egiteko ahalmena dauka. Proiekturako erabilitako hardwareak modu asinkronoa erabiltzera behartzen du, beraz UMSEL00 eta UMSEL01 bitek 0 balioa eduki beharko dute.

   ```
   UCSR0C &=~ (1 << UMSEL00);
   UCSR0C &=~ (1 << UMSEL01);
   ```

2. Paritatea galarazi

   Paritate bitak, USART komunikazio batean hartzaileak jasotako informazioak bere osotasuna mantendu duela egiaztatzeko balio du. Estazio meteorologikoaren kasuan, ez da paritatea erabili eta hori lortzeko UPM00 eta UPM01 bitak zeroko balioa eduki beharko dute.

   ```
   UCSR0C &=~ (1 << UPM00);
   UCSR0C &=~ (1 << UPM01);
   ```

3. Stop bit bakarra

   USART modulua, igorleak bat edo bi stop bit bidaltzeko konfiguratu daiteke. Proiektuan stop bit bakarra erabiliko da, horretarako USBS0 bitak zero izan beharko du.

   ```
   UCSR0C &=~ (1 << USBS0);
   ```

4. Datuen tamaina zehaztu.

Igorle eta hartzaileak transmisioan erabiliko duten datuaren luzera zenbat bitekoa den zehazteko UCSZ0[2:0] bitak erabiliko dira. 3.5 Irudiko taulan ikusi daitekeen bezala, hainbat aukera desberdin daude. Gure kasuan 8 biteko datuak trukatuko dituzte bi nodoek. Hori posible izateko UCSZ00 eta UCSZ01 bitek bateko balioa eduki behar dute eta UCSZ02-k zero. Kontuan eduki behar da UCSZ02 bita UCSR0B erregistroan dagoela (ikusi 3.6 Irudia).

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

3.4. Irudia: UCSR0C erregistroa [atm, 2016]

UCSZ0[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

3.5. Irudia: USART datuaren tamaina [atm, 2016]

UCSR0B

Ondorengoak, 3.6 Irudian ikus daitekeen erregistroko bitak erabiliz egin daitezkeen konfigurazioak zehaztuko dira.

1. Datuen transmisio eta jasotzea gaitu

   Datuen bidalketa eta jasotzea egin ahal izateko, TXEN0 eta RXEN0 bitak 1 balioa izan behar dute.

   ```
   UCSR0B |= (1 << TXEN0);
   UCSR0B |= (1 << RXEN0);
   ```

2. Datuak jasotzeko etenak gaitu

   Datuak inkesta bitartez beharrean etenak erabiliz jasoko dira, modu horretan mikrokontroladoreak beste kode zatiak exekutatzeko aukera izango du eten eskaera bat ez dagoen bitartean. Etenak gaitzeko RXCIE0 bitari bat balioa jarri beharko zaio.

```
UCSR0B |= (1 << RXCIE0);
```

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

3.6. Irudia: UCSR0B erregistroa [atm, 2016]

UDR0

Erregistro hau bi bufferrek partekatzen dute: datu transmisio (TXB) eta jasotze (RXB) bufferrek. Hau da, igorle gisa datua erregistro honetan da eta hartzaile gisa datua erregistrotik irakurtzen da.

Bit	7	6	5	4	3	2	1	0
	TXB / RXB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

3.7. Irudia: UDR0 erregistroa [atm, 2016]

UBRR

USART moduluaren transmisio-abiadura konfiguratzeko UBRR erregistroa erabiliko da. 3.8 Irudian ikusi daitekeen bezala, UBRR erregistroaren balioaren arabera aldatu egingo da transmisio-abiadura. Horretaz gain, abiadura bikoiztu daiteke, UCSR0A erregistroko U2X0 bitaren balioa batekoa izanda.

```
UCSR0A |= (1 << U2X0);
```

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$\text{BAUD} = \frac{f_{\text{OSC}}}{16(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\text{OSC}}}{16\text{BAUD}} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$\text{BAUD} = \frac{f_{\text{OSC}}}{8(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\text{OSC}}}{8\text{BAUD}} - 1$

3.8. Irudia: Transmisio-abiadura edo UBRR erregistroei eman beharreko balioak kalkulatzeko formulak[atm, 2016]

Lortzen diren abiadurak ez dira zehazki lortu nahi direnak, errore txiki bat daukate, UBRR erregistroan jarri beharreko balioak osoak izan behar dutelako. Demagun, 115200 baud-eko transmisio abiadura lortu nahi dugula. UBRR erregistroan jarri beharreko balioa 3.8 irudian agertzen diren formulekin kalkulatuko da. Gure kasuan kalkulua egiteko abiadura bikoiztuarekin kalkulatuko dugu.

$$BAUD = \frac{fosc}{8(UBRR+1)} \rightarrow UBRR = \frac{fosc}{BAUD*8} - 1 = \frac{16000000}{115200*8} - 1 = 16.36 \quad (3.1)$$

3.1 ekuazioan ikus daitekeen bezala emaitza 16.36 da, baina UBRR erregistroan balio osoak jarri behar direnez $UBRR = 16$ izango da, zati hamartarra galduz. Hori dela eta %2.25-eko errore bat sortzen da.

3.3 Tenporizadoreak

Tenporizadoreak baliabide oso erabiliak dira sistema txertatuetan. Hainbat modu desberdinetan funtziona dezakete eta denboraren menpe dauden funtzioen kontrolerako balio dute. Orokorrean tenporizadore bat maiztasun desberdineko seinaleak edo PWM seinaleak sortzeko erabiltzen dira.

Atmega328p tenporizadoreak hiru tenporizadore ditu, *Timer0*, *Timer1* eta *Timer2*. *Timer0* eta *Timer2* tenporizadoreak 8 bitekoak dira, eta kontagailuak eta PWM seinaleak sortzeko bezalako oinarrizko funtzioak erabilgarri dituzte. *Timer1* tenporizadorea berriz, 16 bitekoa da eta funtzioen aldetik beste bien antzekoa da.

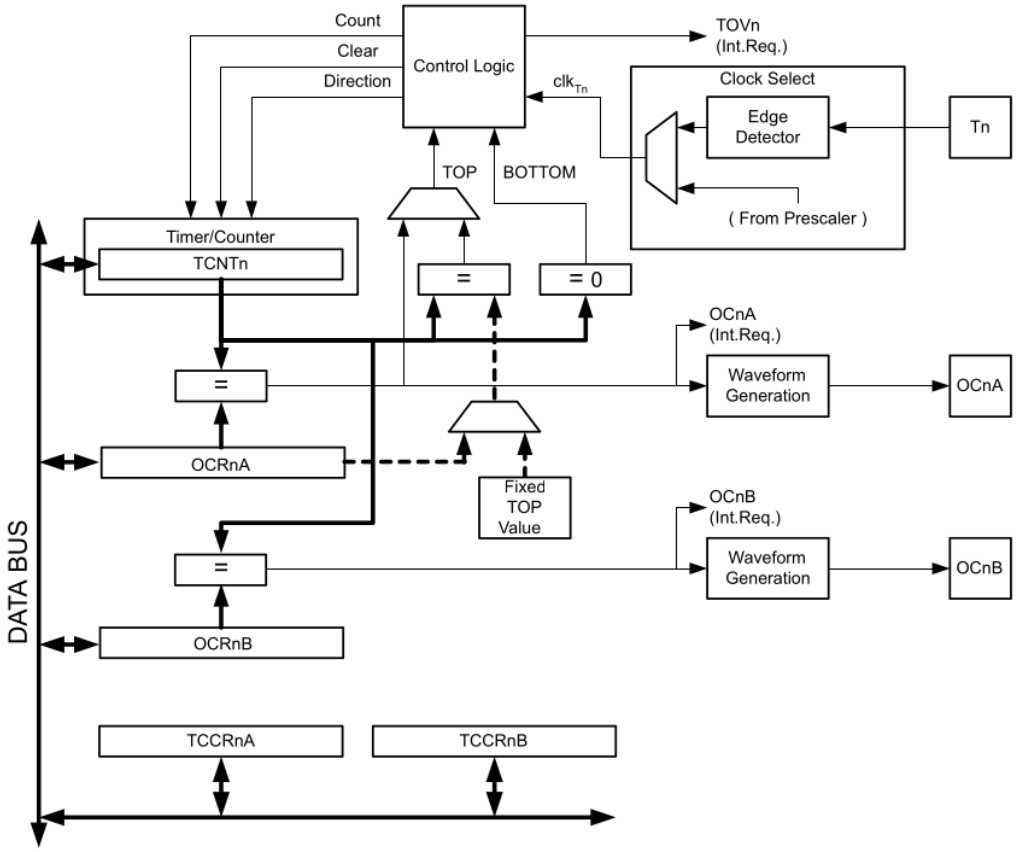
Proiektu honetarako hiru tenporizadoreak erabili dira. *Timer0* DHT-11 sentsorearen erantzun-denbora kontatzeko erabiliko da. DHT-11 sentsorearen erantzuna zehaztutako denbora baino gehiago luzatzen bada, sentsoreari datu eskera berriro egin beharko zaio. *Timer1* tenporizadorea sentsoreei datuak eskatzeko agindua emateaz arduratuko da. Segundu erdiro, sentsore bakoitzaren datuak gordetzen dituzten aldagaiak eguneratu egingo dira. Azkenik, *Timer2* tenporizadorea programa nagusian zehar dauden denbora tarte desberdinak kontrolatzeko erabiliko da.

3.3.1 Clear Timer on Compare match (CTC)

Lehen aipatu bezala, hiru tenporizadoreak erabiliko dira proiektuko ataza desberdinak burutzeko. Tenporizadoreek hainbat operazio modu dituzte erabilgarri, eta horien artean Clear Timer on Compare Watch (CTC) modua aukeratu da hiru tenporizadoreetarako.

CTC motako kontagailuen funtzionamendua simplea da. Kontagailuaren balioa inkrementatzen joango da eta zehaztutako muga batera iristerakoan eten seinale bat sortuko da, eten zerbitzu-errutina exekutatzeko eta kontagailua berriro hasieratuko da. Prozesu berdina behin eta berriz errepikatuko da tenporizadorea gelditu arte.

Atmega328p mikrokontroladoreareko CTC motako kontagailu batean TCNTn eta OCRnA erregistroak eduki behar dira kontuan. TCNTn erregistroa izango da kontagailua eta muga OCRnA erregistroaren bidez zehazten da. Adibide gisa, 3.9 irudiko *Timer0* tenporizadorearen bloke diagraman ikus daitekeen bezala, kontagailuaren balioa (TCNTn) uneoro OCRnA erregistroan gorddetakoarekin konparatzen da seinalea sortzeko.



3.9. Irudia: *Timer0* tenporizadorearen bloke diagrama[atm, 2016]

Eten bat sortu nahi den maiztasunaren arabera OCRnA erregistroan balio bat jarri beharko da. Balio hori ondorengo formularen bitartez kalkulatuko da.

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 * N * (1 + OCRnx)} \tag{3.2}$$

- f_{OCnx}: Lortu nahi den maiztasuna.
- $f_{clk\_I/O}$: Mikrokontroladorearen clock seinalearen maiztasuna.
- N: Prescalerraren balioa.
- OCRnx: Nahi den maiztasuna lortzeko zehaztu behar den balioa.

Adibide moduan, 5ms-ko periodoa daukan tenporizadore bat programatzerakoan OCRnx erregistroak izan beharreko balioa 3.2 formularekin kalkulatuko dugu. Lortu nahi dugun seinalearen maiztasuna 200Hz-koa izango da eta erabiliko dugun preskalerra 256-koa izango da. Balioak 3.2 formulan ordezkatu eta OCRnx baztertu eta gero horrela geratuko lizateke.

$$OCRnx = \frac{16000000}{2 * 256 * 200} - 1 = 156 \tag{3.3}$$

Beraz, OCRnx erregistroaren balioa 156 bada, 5ms-ko periodoa daukan seinale bat sortuko da.

3.3.2 Tenporizadoreen konfigurazioa

Atal honetan hiru tenporizadoreak CTC moduan funtzionatzeko konfigurazioa azalduko da. Hiru konfigurazioak antzekoak dira, beraz, adibide moduan 1 tenporizadorearen konfigurazioa azalduko da.

CTC moduan hasieratu

Tenporizadorea CTC moduan hasieratzeko 3.12 irudian ikus daitekeen taulan zehaztutako bitei eman behar zaizkie balioak. Funtzionamendua aukeratzeko bitak, TCCR1B erregistroko (3.10 Irudia) WGM13 eta WGM12, eta TCCR1A erregistroko (3.11 Irudia) WGM11 eta WGM10 dira. Zehazki *Timer1* tenporizadorearen kasuan hauek dira aurreko bitek hartuko dituzten balioak.

```
TCCR1B &=~ (1 << WGM13);
TCCR1B |= (1 << WGM12);
TCCR1A &=~ (1 << WGM11);
TCCR1A &=~ (1 << WGM10);
```

Bit	7	6	5	4	3	2	1	0
	COM1	COM1	COM1	COM1			WGM11	WGM10
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

3.10. Irudia: TCCR1A erregistroa[atm, 2016]

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

3.11. Irudia: TCCR1B erregistroa [atm, 2016]

Mode	WGM13	WGM12 (CTC1)[1]	WGM11 (PWM11)[1]	WGM10 (PWM10)[1]	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX

3.12. Irudia: *Timer1* tenporizadoreak eskaintzen dituen funtzionatzeko moduak[atm, 2016]

Etenak gaitu

Etenak gaitzeko TIMSK1 erregistroko OCIEA bitari 1 balioa esleitu behar zaio. Modu honetan, eten eskaera bat jasotzen denean zerbitzu errutina exekutatuko du.

Preskalerra

Prescaler batek, osokoen zatiketa baten bidez maiztasun handiko seinale bati maiztasuna txikitzeko balio dute. Hau da, sortutako seinalea balio batekin zatituko du, preskalerraren erregistroak daukan balioaren arabera. Atmega328p txiparen preskalerrak 4 balio desberdin har ditzake: 8, 64, 256 eta 1024. Konfiguraziorako TCCR1B erregistroko CS12, CS11 eta CS10 bitak erabiliko dira. 3.13 Irudian, balio bakoitza aukeratzeko behar diren aurreko biten konbinazioak agertzen dira.

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)

3.13. Irudia: Prescaler desberdinak zehaztapenak[atm, 2016]

Proiektuan zehar, tenporizadore guztien kasuan erabili dira preskalerrak. Adibidez, *Timer1* tenporizadoreak 64-ko preskalerra erabiltzen du, eta *Timer1*-en preskalerra modu horretan konfiguaratzeko ondorengo kodea erabili da.

```
TCCR1B &=~ (1 << CS12);
TCCR1B |= (1 << CS11);
TCCR1B |= (1 << CS10);
```

3.4 TWI (I2C)

Two Wire Interface, I2C bezala ezagutua, bi hankatxo erabiltzen dituen komunikazio protokolo sinple bat da. Hankatxo batetik datua transmitituko da eta bestetik zehaztutako maiztasun bateko clock seinalea. Protokolo honek morroi kopuru handiekin komunikatzeko ahalmena ematen du. Morroi bakoitza identifikatzeko 7 biteko helbideak erabili ohi dira.

Proiektu honetan, bi atmega328p mikrokontroladore erabili dira. Mikrokontroladoreetako bat nagusi bezala jokatuko du eta besteak morroi gisa. Mikrokontroladore nagusia wifi moduluaz, DHT-11 sentsoreaz eta sistema osoaren kontrol orokorraz arduratuko da. Morroia berriz, anemometroaren datuak eskuratzeaz eta, nagusiak eskatzerakoan, bidaltzeaz arduratuko da.

TWI protokoloak hainbat funtzionatzeko modu ditu, edo beste era batean esanda txip nagusi edo morroiek modu desberdinetan joka dezakete.

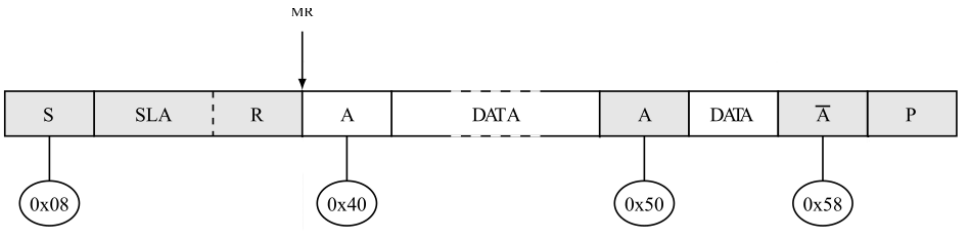
- `Master sender`
- `Master receiver`
- `Slave sender`
- `Slave receiver`

Proiektu honetarako, nagusiak anemometroaren datuak eskatu eta irakurriko ditu (Master receiver) eta morroiak eskatutako datuak bidaliko dizkio (Slave sender).

3.4.1 Master receiver

Atal honetan Atmega328p mikrokontroladore nagusia `Master receiver` moduan konfiguratzeko beharrezkoak diren funtzio eta kontzeptuak landuko dira.

Lehen aipatu bezala nagusiaren papera morroiari datuak eskatzea izango da. 3.14 irudiko diagraman komunikazio osoan zehar nagusiak datu bat jasotzeko sortzen diren egoerak eta mezuak ikus daitezke. Bloke ilunek nagusiak bidaltzen dituen mezuak dira eta argiak morroiarenak.

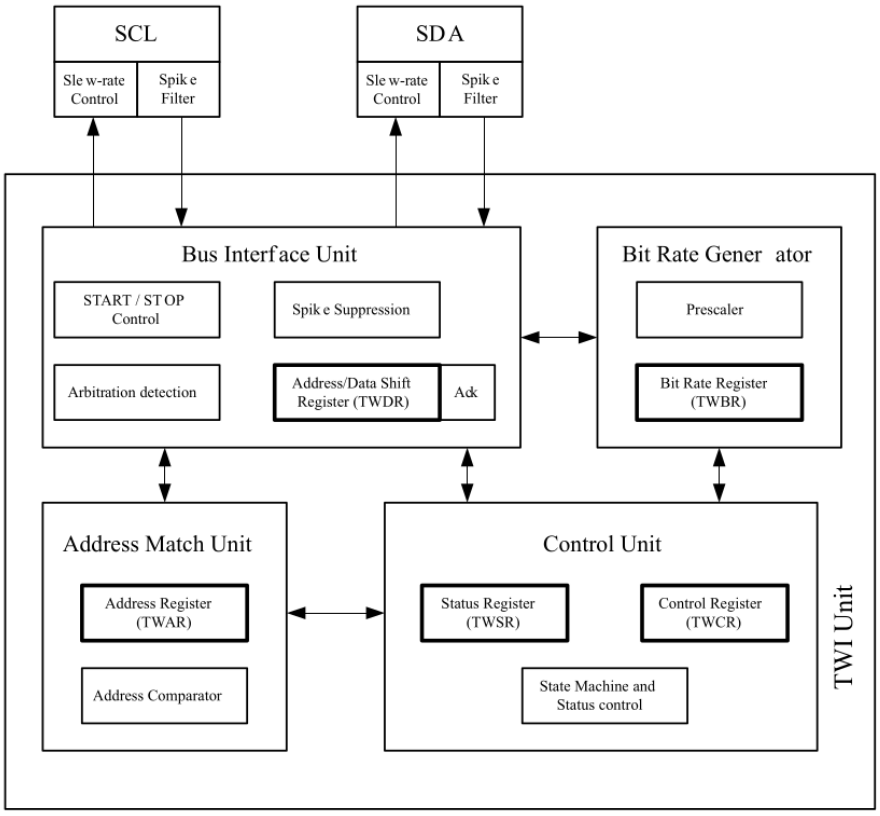


3.14. Irudia: Master receiver moduan datu bat jaso[atm, 2016]

Hasteko morroiari START seinalea bidaliko zaio, komunikazio bat hasi nahi dela adieraziz. Segidan datuak eskatu nahi zaizkion morroiaren helbidea eta datuak irakurtzeko nahia datuen hariaren bitartez bidaliko da. Hau eta gero, morroiak datuak bidaliko ditu eta ACK edo NACK baten bidez adieraziko zaio morroiari datu gehiago irakurri nahi diren ala ez. Amaitzeko nagusiak STOP egoera baten bitartez komunikazioa bukatu dela adierazoko du.

3.15 irudiko diagraman Atmega328p mikorkontroladorearen TWI moduluaren atalak eta erabiliko diren erregistroak ikus daitezke.

- `TWSR (TWI Status Register)`: TWI moduluaren egoera adierazteko balio du. Atmega328p mikrokontroladorearen gidaliburuan [atm, 2016] egoera desberdinen balioak zehazten dira eta balio horiek erregistro honetan jasotakoarekin konparatuta, TWI moduluaren funtzionamendua egokia dela egiazta dezakegu.
- `TWCR (TWI Control Register)`: Erregistro hau TWI moduluaren funtzionamendua kontrolatzeko erabiltzen da. Erregistroko bit desberdinak aktibatuta hainbat portaera lor daitezke.
- `TWAR (TWI Address Register)`: Erregistro honetan morroiaren 7 biteko helbidea gordetzen da.
- `TWDR (TWI Data Register)`: TWI modulua igorle moduan dagoenean, bidali beharreko hurrengo bytea gordetzen du eta hartzaile moduan dagoenean, jasotako azken bytea gordetzen du.
- `TWBR (TWI Bit-rate Register)`: Erregistro honen balioa TWI moduluaren transmisio abiaduraren maiztasuna zehazteko balio du.



3.15. Irudia: TWI moduluaren diagrama[atm, 2016]

Moduluaren hasieraketa

Mikrokontroladore nagusian TWI modulua hasieratzea ataza simplea da, transmisio abiadura zehaztu eta modulua gaitzea baino ez baita behar. Transmisio abiaduraren maiztasuna zehazteko TWBR erregistroa erabiliko da. Morroiaren maiztasunak TWI moduluaren SCL clock seinalearen maiztasuna baino 16 aldiz handiagoa izan behar du gutxienez. Gure kasuan TWI moduluaren maiztasuna 100KHz-koa izango da. TWBR erregistroaren balioa kalkulatzeko ondorengo formula erabili da.

$$f_{SCL} = \frac{f_{clk\_I/O}}{16 + 2 * TWBR} = \frac{16000000}{16 + 2 * 72} = 100000Hz = 100kKz \tag{3.4}$$

3.4 formulan ikus daitekeen moduan 100KHz-ko maiztasuna lortzeko TWBR erregistroak 72 balioa eduki behar du.

Horretaz gain TWI modulua gaitzeko TWCR erregistroko TWEN bitari 1 balioa eman behar zaio.

```
TWBR = 72;
TWCR |= (1 << TWEN);
```

Start eta stop egoerak

TWI protokoloan, komunikazio bakoitza hasi eta bukatzeko start eta stop egoerak sortu behar dira, datu lerroa eta clock seinaleaz baliatuz. Hau guztia kontrolatzeaz mikrokontroladore nagusia arduratuko da uneoro.

Egoera hauek sortzeko prozesuan TWCR erregistroko bit desberdinen konbinazioek hartuko dute parte. Ondoren, TWSR erregistroan jasotzen den kodea espero den kodearekin konparatu beharko da start egoera ondo joan dela egiaztatzeko.

Start egoera bat sortzeko TWCR erregistroko TWINT, TWSTA eta TWEN bitak aktibatu behar dira, eta TWSR erregistroan 0x08 balioa jaso beharko da.

```
TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
//Transmititu arte itxaron
while(!(TWCR & (1 << TWINT)));
//Jasotako erantzuna egiaztatu
while((TWSR & 0xF8) != 0x08);
```

Stop egoera bat sortzeko aldiz, TWCRko TWINT, TWSTO eta TWEN bitek 1 balioa eduki behar dute.

```
TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);
```

Datuen eskaera

Morroiari datuak eskatzeko, morroiaren helbidea eta irakurtzeko agindua TWDR erregistroan idatzi behar dira. Irakurtzeko agindua, morroiaren helbideari 1 erantsiz lortuko da eta idaztekoa 0 erantsiz. Transmisioa burutu arte itxaron ondoren, TWSR erregistroaren bidez dena behar bezala joan dela egiaztatuko dugu, 0x40 kodea jaso dugula begiratuz. Kode horren bidez morroiak eskaera jaso duela eta ACK bat bueltatu duela esan nahiko du.

```
//morroiaren helbidea TWDR erregistroan gorde
int read_addr = 1;
read_addr |= addr << 1;
TWDR = read_addr;
//kontrol bitak
TWCR = (1 << TWINT) | (1 << TWEN);
//Transmititu arte itxaron
while(!(TWCR & (1 << TWINT)));
//Jasotako erantzuna egiaztatu
while((TWSR & 0xF8) != 0x40);
```

Datuak jaso

Morroiak bidaltzen dituen datuak jasotzeko TWCR erregistroko TWINT eta TWEN bitak aktibatuta egon behar dute. Jasotako datua TWDR erregistrotik irakurriko da ondoren aldagai batean gordetzeko. Datua irakurri eta gero beste datu bat jason nahi bada ACK bat bidaliko zaio morroiari TWCR erregistroko TWEA bitari 1 balioa eman behar zaio. Jasotako datua azkenekoa bada berriz,

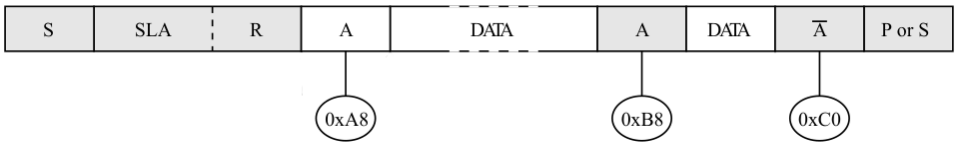
NACK bat bidaliko da TWEA bitari 0 balioa emanda.

```
//Datua jaso eta NACK bidali
TWCR = (1 << TWINT) | (1 << TWEN) | (0 << TWEA);
data = TWDR;
```

3.4.2 Slave Sender

Atal honetan mikrokontroladore morroia slave sender moduan konfiguratzeko beharrezkoak diren funtzio eta kontzeptuak aztertuko dira. Morroiaren kasuan funtzio gutxiago beharko dira, ez baitira start eta stop egoerak sortu behar.

Morroiaren papera, nagusiak eskatzerakoan anemometrotik jasotako datu bat bidaltzea izango da. 3.16 irudiko diagraman nagusiari datu bat bidaltzeko sortzen diren egoerak eta mezuak ikus daitezke.



3.16. Irudia: Slave sender moduan datu bat bidali[atm, 2016]

Hasieran nagusiaren start egoera eta datu baten eskaera iristen dira. Nagusiaren eskaera ondo jaso dela adierazteko morroiak ACK bat bidaltzen du. Jarraian nagusiari datuak bidaliko zaizkio NACK bat jaso arte. Momentu horretan konexioa bukatu egingo da eta beste eskaera berri bat jasotzeko prestatuko da. Proiektu honetan, datu bakarra bidaliko da konexio bakoitzean.

Moduluaren hasieraketa

Morroiak TWI modulua erabili ahal izateko, lehenengo hau hasieratu behar da. Nagusiaren kasuan ez bezala gauza gehiago eduki behar dira kontuan. Hasteko morroiari helbide bat esleituko zaio TWAR erregistroan idatziz. Horretaz gain, TWCR erregistroko TWINT eta TWEN bitak aktibatu behar dira.

```
TWCR = (1 << TWEN) | (1 << TWEA);
TWAR = 0x06 << 1;
```

Azkenik, datuen transmisiorako beharrezkoak diren bi hankatxoak prestatu behar dira. Zehazki sarrera moduan konfiguratuko dira, bestela ez da nagusiak bidalitako eskaera jasotzeko gai izango.

```
DDRC &=~ (1 << PORTC5);
DDRC &=~ (1 << PORTC4);
PORTC = (1 << PORTC4) | (1 << PORTC5);
```

Eskaera jaso

Txip nagusitik eskaera bat jasotzeko, morroia, TWCR erregistroko TWEN, TWINT eta TWEA bitak aktibatuz prestatzen da. TWEA bitaren bidez, nagusiari ACK baten bidez erantzutea ahalbidetzen da. Beharrezko bitak aktibatu eta gero, eskaera bat jaso arte itxarongo da. TWSR erregistroaren bidez, eskaera bat jaso dela eta ACK bat bidali dela kontrolatuko da. Dena behar bezala joan dela egiaztatzeko 0xA8 kodea irakurri behar da TWSR erregistroan.

```
TWCR = (1 << TWEA) | (1 << TWEN) | (1 << TWINT);
//Transmititu arte itxaron
while(!(TWCR & (1 << TWINT)));
//JAsotako erantzuna egiaztatu
while((TWSR & 0xF8) != 0xA8);
```

Datuak bidali

Datu bat bidaltzeko dinamika sinplea da, hasteko TWDR erregistroan behar den datua idatziko da eta ondoren TWCR erregistroko beharrezko bitak aktibatuko dira. Datua transmititu bada eta nagusiaren partez NACK bat jasotzen bada, ez dira datu gehiago bidaliko, baina ez bada horrela TWEA bita ere aktibatu behar da eta beste datu berri bat bidali.

```
TWDR = data;
//NAgusiaren erantzuna NACK bat bada
if((TWSR & 0xF8) == TW_ST_DATA_NACK){
    TWCR = (1 << TWEN) | (1 << TWINT);
```

4. KAPITULUA

Erabilitako sentsore eta moduluak

Kapitulu honetan, estazioa osatzeko erabili diren sentsore eta moduluak agertuko dira. Bakoitzaren ezaugarri nagusiak eta funtzionamendua ezagutaraziko da. Horretaz gain Atmega328p mikrokontroladorearekin, datuak modu egokian jasotzeko erabili diren komunikazio protokolo zein barne moduluak aipatuko dira.

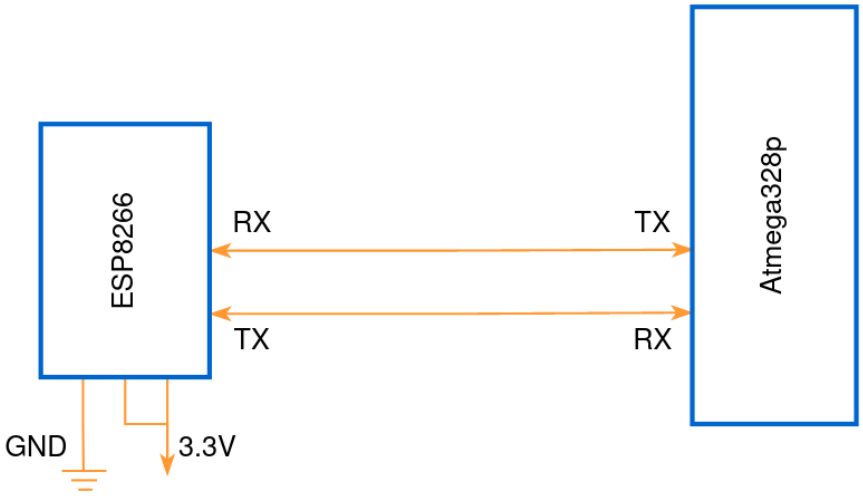
4.1 ESP-8266 Wifi modulua

Datuak bistaratzeko aplikazioaren eta Atmega328p txiparen arteko komunikazioa burutzeko beharrezkoa da ESP8266 Wifi modulua. Modulu hau, *Expressif* enpresak sortutako kostu baxuko txip bat da eta bere erabilera orokorrena mikrokontroladore desberdinei Wifi sare batera konektatzeko eta TCP/IP konexio sinpleak burutzeko ahalmena ematea da.

Atmega328p mikrokontroladoreak ez dauka Wifi sare batera konektatzeko ahalmenik, beraz ESP-8266 Wifi modulu baten laguntza behar du. Bi elementu hauen arteko komunikatzioa UART moduluaren bidez gauzatuko da. Sarean TCP/IP konexioak burutzeko moduluak *Hayes* motako aginduak (AT aginduak) erabiltzen ditu. Moduluaren funtzionamendua zuzena izan dadin, mikrokontroladorearen UART moduluaren transmisio abiadurak 115200 baud-ekoa izan behar du.

4.1.1 Ohiko eskema

Mikrokontroladorearen eta ESP-8266 moduluaren arteko konekxioak 4.1 irudian ikus daitezke. Modulua $3.3V$-rekin elikatzen da eta Atmega328p mikrokontroladorearekin komunikazioa TX eta RX hankatxoen bidez egiten da.



4.1. Irudia: Atmega328p eta ESP8266 konektatzeko eskema

4.1.2 AT agindu multzoa

Agindu multzo hau, *Hayes Communications* konpainiak garatutako lengoaia baten parte da. Bere garaian, agindu multzoa modemak konfiguratzeko erabiltzen zen. Gaur egun, ESP-8266 bezalako gailu askok beren oinarrizko konfigurazioetan daukate barneratuta. ESP8266 moduluak erabiliko dituen aginduak bere aginduen eskuliburuan daude zehaztuta [esp, 2016] [fuho, 2015].

Aginduen formatua

AT agindu bat, hiru elementu nagusik osatzen dute: komandoa, atzizkia eta datua. Komandoak, Wifi moduluari burutu nahi den ekintza adieraziko dio. Atzizkiaren arabera datu bat irakurri edo esleitu daiteke. Agindu guztiak *'AT'* karaktereekin hasi, eta <CR> karakterearekin bukatu behar dira.

```
AT<Komandoa><Atzizkia><Datua><CR>
```

Agindu bakoitza bidali eta gero, ESP-8266 moduluak erantzun bat emango du. Dena ondo badoa, OK bat bidaliko du, eta beharrezkoa bada, eskatutako datuak bidaliko ditu. Arazoren bat baldin badago berriz, ERROR erantzuna bidaliko du.

Moduluaren konfigurazioa

ESP-8266 moduluak hainbat modu desberdinetan egin dezake lan: *Access Point* moduan edo *Station* moduan. Proiektu honetarako ESP-8266 *Access Point* (AP) moduan konfiguratu da, era horretan bezeroak zuzenean modulura konektatuko dira datuak lortu ahal izateko. Moduluak AP gisa lan egin dezan, ondorengoak izan dira erabilitako aginduak.

- AT+CWMODE=2: Moduluari AP moduan lan egiteko adieraziko dio.
- AT+CWSAP=<ssid>,<pasahitza>,<kanala>,<enc>: Agindu honen bidez AP-ren izena, pasahitza, kanala eta enkriptazioa zehazten dira.
- AT+CIPMUX=1: Agindu honek konexio bat baino gehiago egotea gaitzeko balio du.
- AT+CIPSERVER=1,<portua>: Aukeratutako portuan zerbitzari bat hasieratuko da agindu honekin.
- AT+CIPSTO=<denbora>: Agindu honekin timeout bat egoteko denbora zehazten zaio moduluari.

Aurreko aginduekin modulua *Access Point* gisa konfiguratzea lortu dugu, beraz bezeroen eskaerak jasotzeko prestatuta legoke ESP8266 modulua. Bezero bakoitza konektatu egiten denean, datuak eskuratu nahi baditu, estazioari `"GET"` agindua bidali beharko dio. ESP8266 moduluak jasotzen dituen mezuen formatua honako hau da.

```
+IPD,0,6:mezua
```

Hasieran, `"+IPD"` karaktereak jasotzen dira beti parametro desberdinez jarraituta. Lehenengoak erabiltzaile desberdinak desberdinak identifikatzeko balioko du, bigarrena berriz, mezuaren karaktere kopuruaz eta mezuaz osatuta dago. Parametro desberdinak koma baten bidez desberdintzen dira, eta bigarren parametroaren kasuan karaktere kopurua eta mezua bi puntuz daude banatuta.

Wifi moduluak bezeroari erantzuna bidal diezaion, beste AT agindu batez baliatu beharko gara.

```
AT+CIPSEND=id,luzera
```

Lehenengo parametroaren bidez bezeroaren identifikarorea zehaztuko da. Honi esker bezero desberdinek egin ditzakete eskaerak eta bakoitzak behar duen informazioa jaso. Bigarren para-

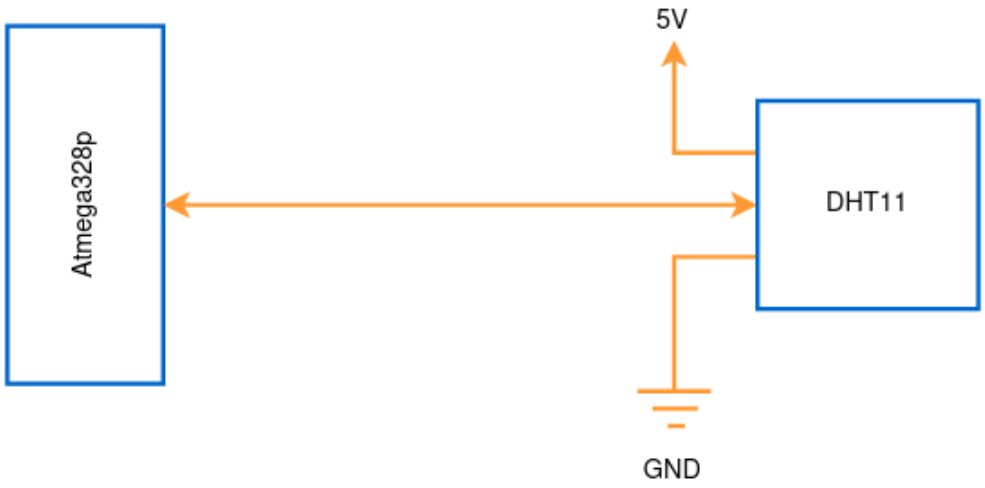
metroaren bidez, bidaliko den mezuaren luzera zehaztuko da. Agindu hau bidali eta gero bidali beharreko mezua zehazten da, lerro salto batez amaituz.

4.2 DHT-11 tenperatura eta hezetasun sentsorea

Tenperatura eta hezetasuna zehaztasun eta sinpletasunez neurtzeko DHT-11 sentsorea erabili da. Sentsorea, 8 biteko mikrokontroladore batek, mota erresistiboko hezetasunerako sentsore batek eta NTC termisore batek osatzen dute. Zirkuitu integratua dagoeneko prestatuta datorrenez, DHT-ak egiten dituen neurketak fidagarriak eta egonkorrak dira. Tenperaturak 0 eta 50 $^{\circ}C$ artean neurtzen ditu eta hezetasuna % 20 eta % 90 artean. NTC termisorearen kasuan sentsibilitatea 2 $^{\circ}C$-koa da eta hezetasun sentsorearen kasuan % 1.

4.2.1 Ohiko eskema

Atmega328p mikrokontroladorea eta DHT-11 sentsorea konektatzeko eskema 4.2 irudian ikus daiteke. Sentsorea 3 eta 5.5V arteko tentsioarekin elikatu daiteke hankatxoetako bat elikadura iturrira konektatuta. Beste hankatxoetako bat mikrokontroladorearen hankatxo batera egongo da konektatuta, datuen trukea egiteko, bi norabidetan. Azkenik, geratzen den hankatxoa lurrera egongo da konektatuta.



4.2. Irudia: Atmega328p eta DHT-11 konektatzeko eskema

4.2.2 Komunikaziorako protokoloa

Komunikaziorako, sentsore honek, bere gidaliburuan [dht,] zehaztutako, hari bakarreko bi noranzkoko serie protokolo espezifiko bat erabili behar da. Bus bakarreko datu formatua erabiltzen da mikrokontroladorearen eta sentsorearen arteko sinkronizazio eta komunikaziorako. Komunikazio prozesu oso batek 4ms inguru irauten du, eta guztira 40 bit bidaltzen dira. Transmisio

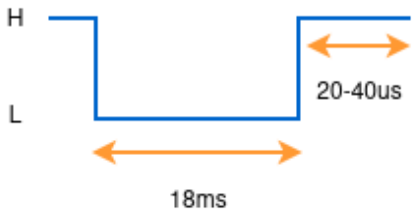
bakoitzean hezetasun sentsorearen zati osoa eta hamartarra, Tenperatura sentsorearen zati osoa eta hamartarra, eta checksum bat bidaltzen dira (4.3 Irudia). Checksuma, datuak ondo jaso direla egiaztatzeko erabiltzen dira. Aurreko 4 datuen batuketa, checksumaren berdina bada datua ondo jaso dela esan nahi du.

4.3. Irudia: DHT-11 sentsoreak bidalitako datu baten formatua

Sentsoretik datuak jaso ahal izateko hainbat pauso jarraitu behar dira. Lehenengo start seinalea bidali behar zaio DHT-11-ri, ondoren sentsoreak hasiera seinale bat bidaliko du eta jarraian datuak.

Start seinalea

Komunikazio bati hasiera emateko agindua da, mikrokontroladoreak sentsoreari datu bat jasotzeko prest dagoela adierazten baitio. Bidali behar den seinaleak ezaugarri jakin batzuk izan behar ditu (4.4 Irudia). Hasteko mikrokontroladorearen hankatxoa irteera moduan jarriko da, eta 18ms bitartean komunikazio buseko tentsioa 0V-koa izango da. Denbora tarte hori pasa eta gero 20 eta 40us bitartean tentsioa altua izatera pasako da. Prozesu hau bukatu eta gero, hankatxoa sarrera moduan jarri behar da, sentsoreak bidaltzen duena jaso ahal izateko.

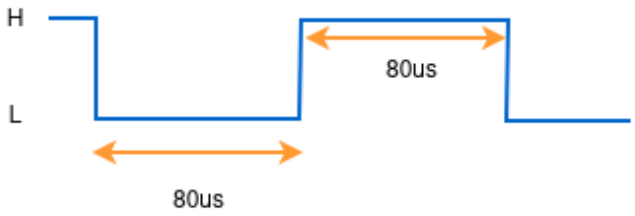


4.4. Irudia: Atmega328p tik bidalitako start seinalea

DHT-11 sentsorearen erantzuna

Start seinalea jasotzen duenean, sentsorea energia aurrezteko modutik exekuzio egoerara pasatzen da, eta mikrokontroladoreari erantzun seinale bat bidaliko dio seinalea jaso duela eta datuak bidaliko dituela adierazteko. Erantzun seinalearen ezaugarriak honako hauek dira (ikus 4.5): Lehenengo 80us bitartean tentsioa nulua izango da busean eta denbora tarte hori bukatzean, beste 80us bitartean tentsioa altua izatera pasako da (4.5 Irudia). Erantzun seinalerako behar den denbora baino gehiago luzatzen bada, komunikazioan arazoren bat egon dela esan

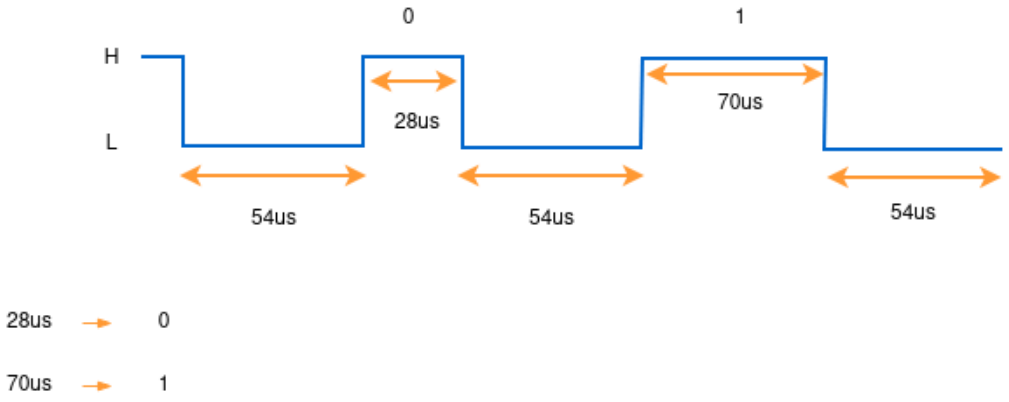
nahiko du. *Timer0* tenporizadorearen bidez, denbora tarte hauek kontrolatuko dira edozein arazo detektatzeko.



4.5. Irudia: DHT-11 sentsorearen erantzun seinalea mikrokontroladoreari.

DHT-11 sentsorearen datu bidalketa

DHT-11 sentsoreak mikrokontroladoreari datuak bata bestearen atzetik bidaltzen dizkio. Datuetako bit bakoitza desberdintzeko 0V-ko 54us irauten duen tarte bat egongo da. Iristen den bitaren balioa tentsio altuko tarteak irauten duen araberakoa izango da. Tarteak 28us irauten badu, bitaren balioa 0 izango da eta 70us inguru irauten badu 1.



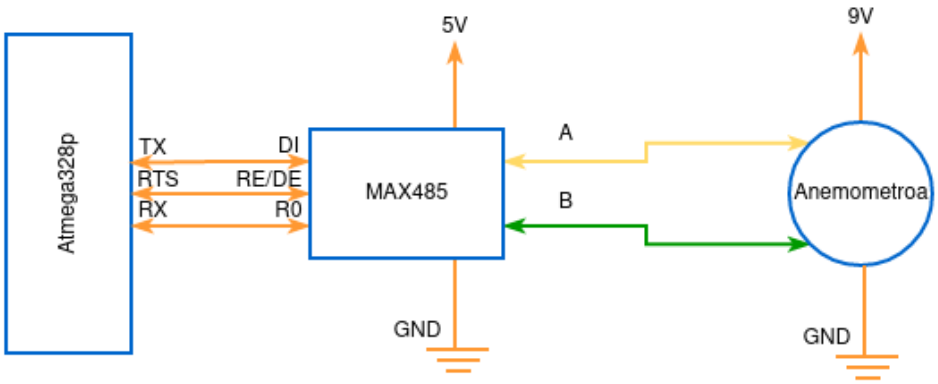
4.6. Irudia: DHT-11 sentsorearen datu bidalketa

4.3 Anemometroa

Anemometroa haizearen abiadura neurtzeko balio duen sentsore bat da. Oso ohikoa da mota honetako sentsoreak estazio meteorologiko batean ikustea. Erabilitako Anemometroa 0 eta 32 m/s arteko abiadurak neurtzeko dago prestatuta. Barnean, hall sentsore bat dauka, anemometroaren zati mugikorrak bira bat ematen duen bakoitzean pultsu elektriko bat sortzeko. Sentsore honek datuen transmisiorako MODBUS protokoloa erabiltzen du. Atmega328p txipak ez dago MODBUS protokoloa ulertzeko prestatuta, beraz, sentsore eta mikrokontroladorearen artean MAX485 modulua jarriko da, datuak MODBUS protokolotik UART protokolora pasatzeko [ane, 2019] [de Craene, 2019].

4.3.1 Ohiko eskema

Atmega328p mikrokontroladorea, MAX485 modulua eta anemometroa konektatzeko egin beharreko lotura guztiak 4.7 irudian ikus daitezke. Anemometroa 9V iturri batera egongo da konektatuta, eta dituen hari horia MAX485-ren A sarrerara eta hari berdea MAX485-ren B sarrerara konektatuko dira. MAX485 modulua bere partez, 5V iturri batez elikatu behar da eta lau hankatxo ditu, mikrokontroladorearekin komunikatzeko. MAX485-ko R0 hankatxoa Atmega328p txipeko RX pinera egongo da konektatuta datuak jasotzeko. DI hankatxoa berriz, TX-ra dago konektatuta, mikrokontroladoretik bidaltzen diren datuak anemometrora berbidaltzeko. Geratzen diren $\overline{RE}$ eta DE, MAX485 modulua datu bat jasotzeko edo bidaltzeko prestatuko dute. $\overline{RE}$ low denean, MAX485 datuak jasotzeko moduan jarriko da eta DE hankatxoa high dagoenean, transmisio moduan. Erosotasunagatik bi hankatxo hauek elkartu daitezke eta Atmega328p txipeko hankatxo bat bakarrik erabiliz kontrola daitezke bi beharrean.



4.7. Irudia: Atmega328p Anemometroa eta MAX485 modulua konektatzeko eskema

4.3.2 Funtzionamendua

MODBUS ingurune industrialean nahiko zabaldua dagoen eta bezero-zerbitzari eredua jarraitzen duen protokoloa da. Nagusira konektatuta dauden morroiei datuak eskatzeko, hainbat metadatuz osatuta dauden tramak bidaltzen dizkiete, eta morroiak eskatutako informazioa beste trama baten barruan bidaliko dio. Elkartrukatzen diren datu-trama hauek 7 eta 8 bytekoak dira. Anemometroaren kasuan bere gidaliburuan [ane, 2019] zehaztuta daude tramako byte bakoitzak hartu behar dituen balioak.

4.8 irudian, anemometroari datu bat eskatzeko trama ageri da. Hasieran morroiaren helbidea eta funtzioaren kodea zehazten dira, ondoren erregistroaren hasierako helbidea eta bere luzera eta azkenik errore-kontrolerako bi byte jartzen dira.

Morroiaren helb.	Funtzioaren kodea	Erregistroa	Erregistroaren luzera	Errore-kontrola	Errore-kontrola
Byte 1	Byte 1	2 Byte	2 Byte	Byte 1	Byte 1
0x02	0x03	0x00 0x00	0x00 0x01	0x84	0x39

4.8. Irudia: Anemometroari datua irakurtzeko eskaera

4.9 irudian berriz, anemometroak aurreko mezuari ematen dion erantzuna ikus daiteke. Lehenengo bi byteak morroiaren helbidea eta funtzioaren kodea adierazten dituzte aurreko kasuan bezala. Ondorengo hiru byteak datuarekin daude erlazionatuta, lehenengo datuaren tamaina zehazten da byte baten bidez, eta gero datua. Azkenik errore-kontrolerako bi byte daude.

Morroiaren helb.	Funtzioaren kodea	Baliozko byte kopurua	Datuak	Errore-kontrola	Errore-kontrola
Byte 1	Byte 1	Byte 1	2 Byte	Byte 1	Byte 1
0x02	0x03	0x02	0x00 0x25	0x3D	0x9F

4.9. Irudia: Anemometroaren erantzun trama

Kontuan eduki behar da, anemometroak bueltatzen duen datua balio erreala baino hamar aldiz handiagoa dela. Beraz, abiadura adierazten duen balioa egokia izateko jasotako datua 10 zenbakiarekin zatituko da. Adibidez, jasotako datua `0x25` da, hamartarrean 37. Haizearen abiadura beraz, $37/10 = 3.7 m/s$ izango da.

5. KAPITULUA

Erabiltzailearen interfazea

Kapitulu honetan, erabiltzaileak estazioak bidalitako datuak ikus ditzan erabiliko duen interfaze grafikoaren funtzionamendua eta ezaugarriak azalduko dira.

Erabiltzaileak estazio meteorologikoak jasotzen dituen datuak ikusi ahal izateko, python lengoaia erabiliz interfaze grafiko bat daukan aplikazioa sortu da. Interneten, estazioaren datuak bistaratzeko hainbat aplikazio aurki daitezke, baina, eskaintzen dituzten baliabideak oso sinpleak dira, eta konplexutasuna igo nahi bada, zerbitzua ez da doakoa. Aplikazio honen kasuan ez du merezi horrelako zerbitzu bat erabiltzea, beraz aplikazioa eskuz garatu da.

Oinarri moduan *Sare Zerbitzu eta Aplikazioak* irakasgaian landutako TCP bezero bat erabili da. Aplikaziorako TCP protokoloa erabiltzea erabaki da, datuen osotasuna bermatu nahi delako.

Aipatu bezala, aplikazioa python programazio lengoaia erabiliz garatu da, eta oinarri moduan hartutako programari hainbat hobekuntza sartzeaz gain, *Tkinter* paketea erabiliz interfaze grafiko bat garatu da, datuak era ikusgarri batean bistaratzeko.

5.1 Kodearen egitura

Aplikazioaren egitura nahiko sinplea da, bi zati nagusi bereizten dira, alde batetik, funtzioak eta bestetik interfazea. Funtzioek aplikazioaren zati logikoa osatzen dute. Interfazearen zatia berriz, beharrezkoak diren elementu grafiko guztiak definitzeko da.

5.1.1 Funtzioak

Funtzioek aplikazioaren zati logikoa osatzen dute, estazioaren ESP8266 Wifi moduluarekin konexio bat mantentzeaz arduratzen baitta zati hori. Ondorengoak dira komunikazioa burutzeko beharrezkoak diren funtzioak.

- `tcp_connect(ip, port)`: Funtzio honen bidez, aplikazioa estazio meteorologikora konektatuko da, socket baten bidez. Funtzioaren hasieran socketa sortuko da eta estazioaren ip helbidea eta portua jakinda konexioa sortuko da.

```
# Socket bat sortu
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Ip helbidea eta portua dauzkan tupla bat sortu
adr = (ip[:-1], int(port))

# Estaziora konektatu
s.connect(adr);
```

- `tcp_send(s)`: Estazioari eskaera egiteaz eta gero datuak jasotzeaz arduratuko da. Estazioa mezu desberdinak interpretatzeko eta hauen arabera zerbitzu desberdinak emateko dago prestatuta. Datuak eskuratzeko "GET"mezua bidali behar zaio estazioari, eta socketa entzuten utzi behar da mezua jaso arte.

```
# Estazioari eskaera bat egin
agindua = "GET\r"
s.send(agindua.encode('utf-8'))
```

  Estazioak ematen duen erantzunaren arabera, hiru egoera desberdin eman daitezke.

– `OK`: Mezua ondo jaso dela adierazten du eta jarraian ":"karaktereaz bananduta neurtutako datuak bidaltzen ditu estazioak.

```
OK:51.0:25.0:25
```

`OK` mezuaren ondoren hezetasunari, tenperaturari eta haizearen abiadurari dagozkien balioak irakurri daitezke hurrenez hurren.

– `ERROR`: Estazioak jaso duen mezua ez badago jaso ditzakeen aginduen zerrendan, errore mezu hau bidaltzen du.

– `TIMEOUT`: Aplikazioak ez badu 10 segundo pasa baino lehen erantzunik jasotzen aplikazioak mezu hau pantailaratzen du erabiltzaileari estazioaren erantzunik jaso ez dela jakinarazteko. Segidan, konexioa eteten da eta gero berriro saiatzen da datuak eskuratzen.

- `tcp_close(s)`: Estazio meteorologikoarekin konexioa eten nahi denean funtzio hau deitzen da.

```
# Estazioarekin konexioa itxi
s.close()
```

5.1.2 Interfazea

Aplikazioaren interfazea garatzeko python lengoaiak daukan *Tkinter* paketea erabili da. Pakete honek interfaze grafikoak eraikitzeko baliabideak eskaintzen ditu. *Tkinter* erabiliz interfaze grafiko bat sortzeko hauek izan dira jarraitutako pausoak.

1. Aplikazioko leiho nagusia sortu.
2. Erabiltzailearen eta aplikazioaren arteko interakzioa egon dadin botoiak edo testu eremuak gehitu.
3. Begizta baten barruan erabiltzaileak sortzen dituen gertaerak kontrolatu, behar den bezala erantzuteko.

5.2 Aplikazioaren deskribapena

Garatutako aplikazioa ez da oso konplexua. Ideia erabiltzaileak aplikazioa ulertzeko erraztasuna izatea da. Horretarako egindako menuak sinpleak dira eta botoi bakoitzaren funtzioa nahiko intuitiboa da. Aplikazioa bi zati nagusitan banatzen da, menua eta datuen eremua (5.1 Irudia).

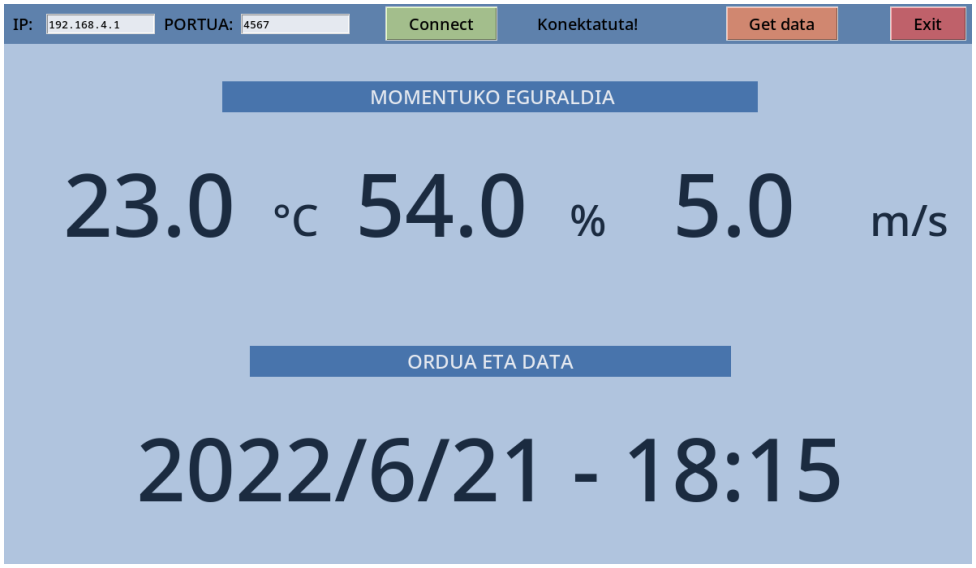
5.2.1 Menu nagusia

Menu honen bidez, erabiltzaileak aplikazioaren funtzioa erabiltzeko gaitasuna izango du. Aplikazioa hasieratu bezain pronto, menuko ezkerraldean dauden eremuak beteko dira zerbitzariaren IP helbidea eta portua zehaztuz. Lehenetsitako 192.168.4.1 helbidea eta 4567 portua daude jarrita. Ondoko *Connect* botoia sakatuz, `tcp_connect` funtzioa exekutatuko da eta estaziora

konektatuko da. Botoi honen alboan dagoen hutsunean mezu desberdinak agertuko dira konexioari buruzko mezuak bistaratuz, arazoen jarraipena errazteko. Hau egin eta gero, *Get data* botoia sakatu daiteke, `tcp_send` funtzioaren bidez estazioari datuen eskaera egiteko. Botoia sakatu eta gero, `tcp_send` funtzioa bost segundoro exekutatuko da. Azkenik, *Exit* botoia sakatzerakoan konexioa eten egingo da eta aplikazioa itxiko da.

5.2.2 Datuen eremua

Interfazearen zati honetan estazio meteorologikoak bidalitako datuak bistaratuko dira. Goiko aldean estazioak bidalitako tenperatura, hezetasuna eta haizearen abiadura bistaratzen dira, eta beheko aldean data eta ordua ikusi daitezke. Datuak bistaratzeko, aplikazioa estazioarekin konektatuta egon behar du eta menuko *Get data* botoia sakatu behar da.



5.1. Irudia: Datuak bistaratzeko aplikazioaren interfaze grafikoa

5.3 Estaziora konektatzeko beste moduak

Estazio meteorologikora konektatzeko eta uneko datu meteorologikoak lortzeko, sortutako interfazeaz gain, beste hiru modu desberdin daude. Atal honetan modu alternatibo hauen azalpen labur bat emango da.

5.3.1 Interfaze grafikorik gabeko aplikazioa

Aurreko aplikazioaren agindu lerroko bertsioa da (5.2 Irudia). Erabilera sinplea da, aplikazioa exekutatu eta gero, "`[SEND COMMAND]`" testuaren azpian estazioari bidaliko zaion agindua idatzi behar da eta enter tekla sakatu.

```
[unai@fedora app]$ python3 tcp_client.py 4567
[SEND COMMAND]
> GET
[SERVER RESPONSE]
OK:51.0:25.0:0

[SEND COMMAND]
> exit
```

5.2. Irudia: Interfaze grafikorik gabeko aplikazioaren erabilera

Irudian ikus daitekeen bezala, lehenengo `[SEND COMMAND]` azpian, `GET` agindua bidaltzen zaio estazioari. Azpian estazioak bidalitako erantzuna ageri da, hezetasunaren, tenperaturaren eta aizearen abiaduraren balioekin. Azkenik `exit` aginduaren bitartez konexioa eteten da.

5.3.2 Netcat

Netcat, TCP edo UDP protokoloak erabiliz sarean idazteko edo irakurtzeko balio duen baliabide bat da. Debianen oinarritutako sistema eragileetan, hurrenko komandoaren bidez instalatu daiteke.

```
$ sudo apt install netcat
```

Baliabide honek mota askotako konexioak sortzeko aukerak eskaintzen ditu, baina gure kasuan TCP bezero sinple bat baino ez dugu erabiliko. Hau sortzeko hurrengo agindua exekutatu behar da zerbitzariaren helbidea eta portua parametro moduan zehaztuz.

```
$ nc 192.168.4.1 4567
```

Agindua exekutatu eta gero terminalean `GET` komandoa idatzi behar da estazioak eskatutako datuak bidaltzen dituela egiaztatzeko. 5.3 irudian, `GET` agindua jaso eta gero sentsoreen informazioa agertzen dela ikusi daiteke, horretaz gain `GET` ez diren aginduak bidaltzean errore mezua bueltatzen dela ikusi daiteke ere.

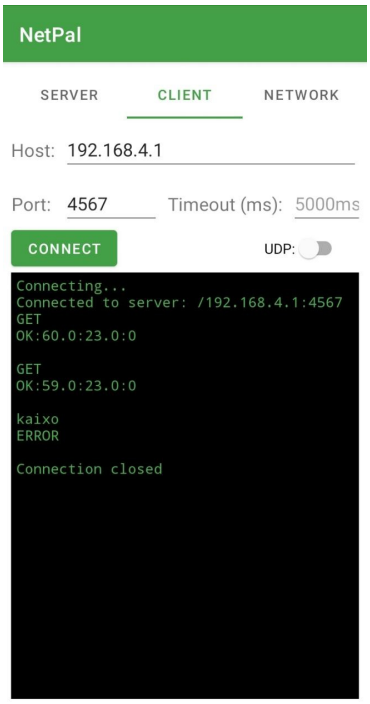
```
[unai@fedora app]$ nc 192.168.4.1 4567
GET
OK:46.0:23.0:0
GET
OK:46.0:23.0:0
a
ERROR
kaixo
ERROR
GET
OK:53.0:23.0:0
```

5.3. Irudia: Netcat baliabidearen erabilera estazioaren datuak irakurtzeko

Datuen informazioa jasotzeaz gain, adibide moduan `"a"` eta `"kaixo"` aginduak bidali dira, estazioak errore mezua bueltatzeko.

5.3.3 NetPal

Ordenagailu bat erabiltzeaz gain NetPal aplikazioaz baliatuz telefono mugikor bat erabili daiteke estazio meteorologikora konektatzeko. NetPal, Netcat baliabidearen antzera funtzionatzen duen aplikazio bat da telefono mugikor batean erabiltzeko. 5.4 irudian ikusi daitekeen bezala *Host* eremuan IP helbidea zehaztuko da eta azpiko *Port* eremuan portua. *Connect* botoia sakatu eta gero azpiko eremu beltzean mezu bat agertuko da konektatu dela adieraziz. Hortik aurrera erabiltzaileak estazioari datuak eskatu ahal dizkio.



5.4. Irudia: NetPal aplikazioaren erabilera

Aurreko proben antzera, `GET` mazua bidali da datuak eskuratzeko eta errore mazu bat jasotzeko `kaixo` agindua. Azkenik *Disconnect* botoia sakatuz estazioarekin konexioa etengo da.

6. KAPITULUA

Sistemaren garapena

Kapitulu honen helburua, estazioa osatzen duten bi mikrokontroladoreen programa nagusiak aztertzea eta aurretik landutako sentsore eta moduluen programekin dauzkaten loturak azaltzea da. Kapituluan zehar landuko den kode guztia github-en[1] egongo da eskuragarri.

[1] https://github.com/UnaiFernandez/minimeteo

Hiru eta lau kapituluetan 6.1 irudiko estazio meteorologikoa osatzen duten modulu eta sentsore bakoitza aztertu da. Atal honen helburua modulu bakoitza kontrolatzeko erabili den kodea estazioaren kontrol orokorrerako nola erabili den azaltzea da.

Proiektuko kodea bi atal nagusitan banatu da, lehenengoa mikrokontroladore nagusia kontrolatzeaz arduratuko da eta bigarrena morroia kontrolatzeaz. Segidan mikrokontroladore bakoitzaren programa nagusia azaldu egingo da programa honen, eta sentsore eta moduluen artean dauden loturak aztertzeko.

6.1. Irudia: Estazio meteorologikoa

6.1 Mikrokontroladore nagusia

Mikrokontroladore nagusia izango da garrantzi handiena edukiko duen gailua. Wifi modulua kontrolatzeaz, tenperatura eta hezetasuna lortzeaz eta morroiarekin komunikatzeaz arduratuko da. Proiektuan erabili diren sentsore eta modulu bakoitzaren kodea eta funtzioak fitxategi desberdinetan banatu dira, eta denak bateratzeaz eta erabiltzeaz main.c fitxategi nagusia arduratuko da. Honako hauek dira mikrokontroladore nagusirako erabiltzen diren fitxategiak:

- `main.c`
- `USART.c`
- `wifi.c`
- `DHT11.c`
- `timers.c`
- `twi_master_receiver.c`

Segidan estazioaren kontrol orokorraz arduratuko den `main.c` programaren azterketa egingo da. Hasteko USART eta TWI moduluak, eta LEDak pizteko beharrezkoak izango diren hankatxoak hasieratuko dira. `Timer2` tenporizadorea hasieratuko da ere, 100ms iraungo duen tarte bat itxaroteko.

```
//UART modulua hasieratu 115200 baud-etara
init_USART(115200);
//TWI hasieratu
init_TWI();
//etenak gaitu
sei();
//LED-ak hasieratu;
init_LED();
```

Hasieraketak eta 100ms pasa eta gero, wifiaren konfigurazioa hasiko da. Konfigurazioarako, 6.1.1 atalean definituta dauden funtzioak erabiliko dira. Wifiaren konfigurazioa ondo badoa, LED berde bat piztuko da erabiltzaileak ikus dezan.

```
int start = 1;
//Wifi modulua hasieratu
if(!hello_ESP())
    start = 0;
//Operazio modua aukeratu
if(!ESP_mode(AP))
    start = 0;
//AP-aren konfigurazioa
if(!AP_setup("MINIMETEO_v.1", "12345678", 1, 4))
    start = 0;
//Konexio anitzak gaitu
if(!ESP_multiple_conn(1))
    start = 0;
//Zerbitzaria hasieratu 4567 portuan
if(!ESP_server(1,"4567"))
    start = 0;
//Zerbitzariaren timeouta ezarri
if(!ESP_server_timeout("120"))
    start = 0;

if(start == 1){
    PORTB |= (1 << PORTB4); //LED berdea piztu
```

Wifiaren konfigurazioa ondo joan baldin bada, `Timer1` tenporizadorea hasieratu egingo da, segundu erdiro sentsoreei datuak eskatzeko. Honen segidan, programaren begizta nagusia hasten da, bertan sentsoreei datuak noiz eskatu eta bezeroari noiz bidaliko zaizkion kontrolatzen da.

```
//Timer1 hasieratu
init_timer1();

while(1){
    ...
}
```

Bezeroak "GET" mezua bidaltzen duenean, USART moduluak eten bidez jaso egiten du eta get_command aldagaian gordetzen du jasotako mezua. Mezua jaso dela adierazteko, send_msg aldagaiari 1 balioa ematen zaio, eten zerbitzu-errutinan, gero begizta nagusian dagoen TCP_response funtzioa exekuta dadin. Funtzio horren bitartez, jasotako komandoaren kontrola egiten da eta bezeroari bidali beharreko mezua prestatzen da.

```
while(1){
    ...
    if(send_msg == 1){
            USART_string(get_command);
            TCP_response(get_command);
            send_msg = 0;
    }
    ...
}
```

Datuak sentsoreei eskatzeko tarteak *Timer1* tenporizadoreak zehaztuko ditu. Tenporizadorea segundo erdiro eteteko dago prestatuta, hau da sentsoreei datuak segundo erdiro eskatzen zaizkio. *Timer1*-en zerbitzu errurina barruan funtzioak ez exekutatzeko get_data eta get_anem aldagaiei 1 balioa ematen zaie.

get_data aldagaiak 1 balioa daukanean DHT-11 sentsorearen datuak eskatuko dira get_dht_data funtzioa erabilita. Funtzio honek, DHT-11 modulurako definitutako funtzioak ordenean erabiliko ditu datu bat eskatzeko.

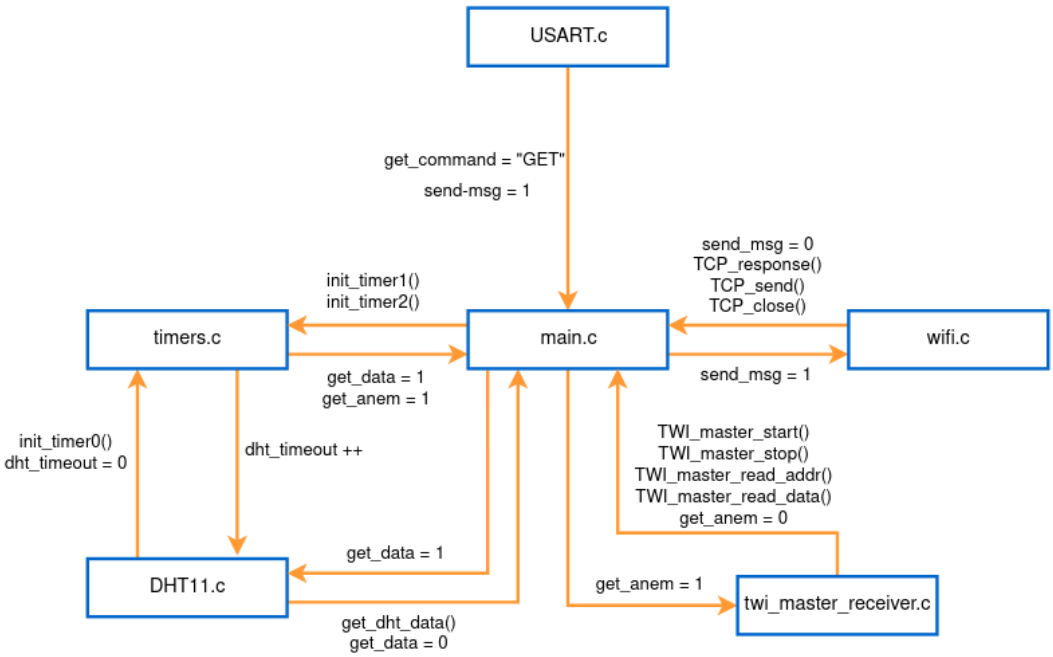
```
while(1){
    ...
    if(get_data == 1){
        //Tenperatura eta hezetasuna lortu
        get_dht_data();
        get_data = 0;
    }
    ...
}
```

get_anem aldagaia berriz anemometroaren datuak lortzeko balioko du. Horretarako TWI moduluaren bidez morroiarekin komunikatuko da.

```
while(1){
    ...
    if(get_anem == 1){
            //Anemometroaren datuak lortu
            TWI_master_start();
            TWI_master_read_addr(0x06);
            anem[i] = TWI_master_read_data(0);
            TWI_master_stop();
```

```
            i++;
            if(i > 3){
            get_anem = 0;
            i = 0;
            }
        }
    }
```

6.2 irudian dagoen diagraman, azaldu berri diren, mikrokontroladore nagusian parte hartzen duten prozesu guztien laburpena ikus daiteke.



6.2. Irudia: Mikrokontroladore nagusiaren programaren diagrama ([atm, 2016])

6.1.1 Funtzioen deskribapena

Atal honetan, txip nagusiko fitxategi bakoitzean aurki daitezkeen funtzioen deskribapen txiki bat emango da.

USART.c

- `init_USART(long int baud)`: funtzio honen bitartez USART modulua konfiguratu egingo da. Parametro bezala transmisio-abiadura eskatzen du. Konfigurazioa egin ahal izateko aurreko atalean aztertutako bitak erabiltzen dira.
- `USART_tx(char d)`: Parametro moduan sartzen den karakterea transmititzeko funtzioa.
- `USART_string(char * string)`: Funtzio honek sartutako karaktere katea transmititu egiten du. Karaktere bakoitza bidaltzeko *USART_tx(char d)* funtzioa erabiltzen du.

- `USART_RX_vect`: Zerbitzu errutina hau jasotzen diren datuen prozesamenduaz arduratuko da. Iristen den karaktere bakoitzeko exekutatuko da errutina eta karaktereak bektore batean bilduko ditu mezu bat osatu arte. Bi mezu mota aztertuko dira eten zerbitzu errutina honetan, alde batetik, Wifi moduluak mikrokontroladoretik transmititutako mezuei emandako erantzunak, eta bestetik, bezeroak bidaltzen dituen eskaerak.

```
    ISR(USART_RX_vect){
            tmp_buff = UDR0;

            //Jasotako mezuaren lehen letra 'O' edo 'E' bada, komando baten
            //responsea dela esan nahi du.
            if(tmp_buff == 'O' || tmp_buff == 'E' || tmp_buff == '+' ||
                resp_first == 1){

            //ESP-01 moduluak bidaltzen dituen karaktereak response
                bufferrean jaso
                response[resp_index] = tmp_buff;
                resp_index++;
            resp_first = 1;

        //Iritsitako mezuaren bukaera detektatu
            if(resp_index == BUFF_SIZE || tmp_buff == '\r' || tmp_buff ==
                '\n'){
                    resp_index = 0;
                    resp_first = 0;

                //Jasotako mezua "+IPD" karaktereekin hasten bada, TCP mezu
                    bat dela esan nahi du.
                if(strstr(response, "+IPD") != NULL){
                    PORTB |= (1 << PORTB3); //LED horia piztu
                send_msg = 1;

                //Jasotako mezua array batean gorde
                strcpy(get_command, response);
                }
            }
    }
}
```

timers.c

- `init_timer0`: *Timer0* tenporizadorea hasieratzeko funtzioa. Aurreko atalean azaldutako bitak erabiltzen dira konfigurazioa egiteko. 10 mikrosegunduro eten eskaera bat egiteko dago prestatuta.
- `stop_timer0`: *Timer0* tenporizadorea gelditzeko funtzioa.
- `TIMER0_COMPA_vect`: *Timer0* tenporizadorearen eten zerbitzu errutina. DHT-11 sentsorearen erantzun denbora handitzen doa, 10 mikrosegunduro, errutina exekutatzen den

bakoitzean.

- `init_timer1`: *Timer1* tenporizadorea hasieratzeko funtzioa. Aurreko atalean azaldutako bitak erabiltzen dira. Segunduro eten eskaera bat egingo du.
- `TIMER1_COMPA_vect`: *Timer1* tenporizadorearen eten zerbitzu errutina. Segunduro exekutatuko da eta sentsoreetatik datuak jasotzeko aldagaia gaituko du.

wifi.c

- `send_command`: ESP8266 modulura aginduak bidaliko dira eta hauen erantzuna itxarongo da funtzio honen bidez.
- `hello_ESP`: Wifi moduluaren funtzionamendu egokia egiaztatzeko funtzioa.
- `AP_setup`: Funtzio honek, ESP8266 Access Point moduan konfiguratzeko balio du.
- `ESP_mode`: Funtzio honen bidez, moduluaren operazio modua aukeratuko da.
- `ESP_multiple_conn`: Konexio bat baino gehiago gaitzeko edo desgaitzeko balio du funtzio honek.
- `ESP_server`: Funtzio hau ESP8266 modulua, zerbitzari moduan hasieratzen du.
- `ESP_server_timeout`: Zerbitzariaren erantzun denbora mugatzeko funtzioa.
- `TCP_send`: TCP protokoloaren bidez, parametro moduan zehazten den mezua bezeroari bidaltzeko.

```
int TCP_send(int id, char* msg){
    char command [24] = "\0";
    int size = strlen(msg)+2;

    sprintf(command, "AT+CIPSEND=%d,%d", id, size);
    t2_count = 0;
    delay_ms(100);
    if(send_command(command, "OK") == RESPONSE_OK){
        //mezua bidali
        USART_string(msg);
        USART_string("\n\r");
        return 1;
    }else{
        PORTB |= (1 << PORTB5); //LED gorria piztu
    }
    return 0;
}
```

- `TCP_response`: Wifi moduluak jaso duen mezuaren arabera, erantzun bat prestatzen du eta `TCP_send` funtzioa erabiliz bidaltzen dio bezeroari.

```
int TCP_response(char * msg){
    //GET hitza badauka mezua bidali bestela errore mezua bidali
    t2_count = 0;
```

```
    int conn_id = msg[5]-'0';

    if(strstr(msg, "GET") != NULL){
        char m[22] = "\0";
        char h_1[4];
        char h_2[4];
        char t_1[4];
        char t_2[4];


        itoa(hezetasuna[0], h_1, 10);
        itoa(hezetasuna[1], h_2, 10);
        itoa(tenperatura[0], t_1, 10);
        itoa(tenperatura[1], t_2, 10);

        sprintf(m, "OK:%s.%s:%s.%s:%s", h_1, h_2, t_1, t_2, anem);
        TCP_send(conn_id, m);
        return 1;
    }else{
        TCP_send(conn_id, "ERROR");
    }
    return 0;
}
```

- `TCP_close`: TCP konexio bat ixteko balio du funtzio honek.

DHT11.c

- `dht_timeout_error`: Timeout errore bat baldin badago datuak eskuratzeko orduan, berriro prestatzen du txipa `start` seinalea bidaltzeko.
- `dht_init`: Mikrokontroladorean aukeratutako hankatxoa prestatu egiten du DHT-11 sentsoretik datu bat irakurtzeko.
- `dht_start`: DHT-11 sentsoreari `start` seinalea bidaltzen zaio.
- `dht_response`: Funtzio honen bidez, `start` seinalea bidali eta gero sentsoreak ematen duen erantzunari itxaroten zaio.
- `dht_data`: Funtzio honek, sentsoreak bidalitako seinalea interpretatzeko eta bertatik hezetasun eta tenperaturaren zenbakizko balioak ateratzeko balio du.
- `dht_checksum`: Funtzio hau, datua ondo jaso dela egiaztatzeko erabiliko da.
- `get_dht_data`: Aurreko funtzio guztiak exekutatzen ditu behar den ordenan.

```
void get_dht_data(){
    //DHT11 modulua hasieratu
    dht_init();
    //Start seinalea bidali
    dht_start();
```

```
        //Erantzuna itxaron
        dht_response();

        //Datu berriak gorde
        if(new_data == 1){
            hezetasuna[0] = dht_data();
            hezetasuna[1] = dht_data();
            tenperatura[0] = dht_data();
            tenperatura[1] = dht_data();
            checksum = dht_data();
            //Egiaztapena
            dht_checksum(hezetasuna[0], hezetasuna[1], tenperatura[0],
                tenperatura[1], checksum);
            new_data = 0;
        }
    }
```

twi_master_receiver.c

- `init_TWI`: TWI modulua hasieratzeko funtzioa
- `TWI_master_start`: Funtzio honen bidez morroiari `start` egoera bat bidaliko zaio.
- `TWI_master_stop`: Funtzio hau morroiari `stop` egoera bat bidaltzeko erabiltzen da.
- `TWI_master_read_addrunsigned char addr`: Funtzio honekin, nagusiak morroiari datu bat irakurri nahi duela adieraziko dio.
- `TWI_master_read_dataint ack`: Morroiak bidalitako datuak irakurtzen ditu.

6.2 Mikrokontroladore morroia

Mikrokontroladore morroiaren betekizunak, anemometrotik datuak eskuratzea eta, nagusiak eskatzen duenean, hauek bidaltzea izango da. Mikrokontroladore nagusian ez bezala, morroiak programa nagusi bat dauka bakarrik, `twi_slave_sender.ino`.

Anemometroarekin UART bidez komunikatzeko arazoak direla eta, programaren zati hori Arduino inguruneak eskaintzen duen `serial.h` liburutegia erabili da. Liburutegia erabili ahal izateko morroiaren kodea Arduinoarentzako programa batean bildu da.

Arduino inguruneko programa batek, bi funtzio nagusik egituratzen dute, `setup` eta `loop`. `setup` funtzioa hasieraketak egiteko erabiltzen da eta `loop` funtzioa begizta nagusian exekutatu beharrekoa zehazteko. Bi funtzio horietaz gain TWI modulua *slave sender* moduan konfiguratzeko eta kontrolatzeko beharrezkoak diren funtzioak aurki daitezke.

Programako `setup` funtzioan anemometroarekin komunikatzeko UART modulua hasieratuko da, `serial.h` liburutegiko `Anem.begin()` erabilita, eta TWI modulua hasieratuko da *slave sender* moduan.

```
void setup() {
```

```
    //Anemometroa hasieratu
    pinMode(RTS_pin, OUTPUT);
    Anem.begin(9600);
    delay(2000);
    //I2C modulua hasieratu
    init_TWI_slave();
}
```

Begizta nagusian, `TWI_slave_write_match` funtzioaren bidez, nagusiaren eskaera detektatzen da eta `TWI_slave_write_data` erabiliz bidaliko zaio datua nagusiari, karakterez karaktere. Datu osoa bidaltzen denean `get _anem` funtzioaren bidez eskatuko zaio datu berri bat anemometroari.

```
void loop() {
    static int i = 0;
    static char msg [3] = "00";

    //Eskaera topatu
    TWI_slave_write_match();
    //Datua bidali
    TWI_slabe_write_data(msg[i]);

    if(i < 3){
        i++;
    }else{
        i = 0;
        //Datu osoa bidali denez, berri bat eskatzen zaio anemometroari
        get_anem(msg);
    }

    delay(100);
}
```

6.2.1 Funtzioen deskribapena

Atal honetan twi_slave_sender.ino fitxategian aurki daitezkeen funtzioen azalpena aurkitu daiteke.

- `init_TWI_slave`: Funtzio honekin, morroiak TWI modulua hasieratzen du.
- `TWI_slave_write_match`: Funtzio honen bidez, nagusiaren eskaera detektatuko da.
- `TWI_slave_write_data(unsigned char data)`: Funtzio hau nagusiari datuaren karaktere bat bidaltzeko balio du.
- `get_anem`: Anemometroaren datuak eskuratzeko funtzioa.
- `setup`: Arduino ingurunean, hasieraketak egiteko prestatuta dagoen funtzioa
- `loop`: Arduino ingurunean, begizta nagusian exekutatu nahi den kodea bertan jartzen da.

7. KAPITULUA

Ondorioak

Azken kapitulu honetan, egindako proiektuari buruzko hausnarketa bat egingo da eta etorkizunerako egingo diren hobekuntza batzuk azalduko dira.

Proiektu honetan, Atmega328p mikrokontroladorearen funtzionamendua aztertu egin da eta estazio meteorologiko erabilgarri bat eraikitzea lortu da. Datuak eskuratzeko hainbat sentsore konektatu dira eta mikrokontroladoreen eta sentsoreen arteko komunikazioa protokolo desberdinak erabiliz burutu da. Horretaz gain, proiektua osatzen duen interfaze grafikodun aplikazio bat gehitu da datuak bistaratzeko.

Garapenean zehar Atmega328p mikrokontroladorea nahiko simplea edo oinarrizkoa iruditu zait baliabideen aldetik. Esaterako, proiektuan wifia eta anemometroaren datuak lortzeko bi UART modulu behar dira, eta mikrokontroladore honek bakarra baino ez dauka. Aurkitutako irtenbidea bi Atmega328p erabiltzea izan da eta biak TWI (I2C) erabilita komunikatzea.

Proiektuaren garapenean zehar hainbat arazo topatu dira, batez ere TWI moduluaren garapenean zehar. Zehazki, morroiarekin komunikatzeko zailtasunak egon ziren, hasieratu eta gero ez baitzituen nagusiaren eskaerak jasotzen. Arazoa konpontzeko, mikrokontroladorearen gidaliburuan begiratzeaz gain, Arduinok eskainitako TWI liburutegiaren kodea aztertu behar izan zen. Arazoen konponbidea topatzea zaila izan arren, irtenbide bat ematea lortu zen behar zen denbora baino gehiago eskaini gabe.

Orokorrean proiekturako zehaztutako helburu guztiak bete dira. Atmega328p mikrokontroladorea erregistro mailan eta Arduino liburutegiak ahal den neurrian erabili gabe programatzea lortu da. Horretaz gain, hainbat sentsore eta modulu aztertu eta erabili dira datuak jasotzeko. Azkenik, *Sistema Txertatuen Oinarriak* eta *Sare zerbitzu eta Aplikazioak* irakasgaietako kontzeptuak proiektuan aplikatzea lortu da.

7.1 Etorkizunerako lana

Azken atal honetan, bukatzeko geratu diren edota etorkizunean proiektua hobetzeko asmoz gehitu daitezkeen hobekuntzak komentatuko dira.

- **Anemometroaren datuak eskuratzeko kodea erregistro mailan** jartzea. Zati honen garapena ez da behar bezala funtzionatzea lortu, USART moduluak ematen duen arazoa ez delako topatu. Horregatik Arduino inguruneak eskaintzen duen `SoftwareSerial.h` liburutegia erabiliz garatu da.
- **Sare protokoloaren segurtasuna** ez da kontuan hartu proiektuan, eta mezu guztiek hutsean bidaiatzen dute sarean zehar. Hobekuntza moduan mezu hauek kriptografia bidez, edo beste segurtasun baliabideekin babestea izango litzateke ideia.
- **Sentsore gehiago** erabiliz, estazio meteorologikoa osatu. Estazioak tenperatura, hezetasuna eta haizearen abiadura bezalako ohiko datuak jasotzen ditu. Horri, plubiometro bat, haize-orratz bat edo presio atmosferikoa neurtzeko sentsoreak gehituta, bezeroak jasoko lukeen informazioa osatuagoa egongo litzateke.
- **Aplikazioa hobetzea** izango litzateke beste aukera bat. Garatutakoa nahiko sinplea da eta datuak bistaratzeko baino ez du balio. Jasotako informazioa datu base batean gordeta edo estaziora agindu desberdinak bidaliz aplikazioa konplexuago bat lor daiteke, modu horretan, datuak bistaratzeaz gain datuen historikoa bistaratzeko adibidez.

Bibliografia

[dht,] *DHT11 Humidity & Temperature Sensor*. OSEPP Electronics. Translated version.

[atm, 2016] (2016). *Atmega328P datasheet complete*. Atmel.

[esp, 2016] (2016). *ESP8266 AT Instruction Set*. Espressif Systems IOT Team.

[ane, 2019] (2019). Rs485 wind speed transmitter sku sen0483. https://wiki.dfrobot.com/RS485_Wind_Speed_Transmitter_SKU_SEN0483. (2022-05-25).

[de Craene, 2019] de Craene, P. (2019). How to build an anemometer with a rs485 modbus win sensor for arduino and esp8266. (2022-05-25).

[fuho, 2015] fuho (2015). Esp8266 - at command reference. https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/. (2022-03-08).