UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Vlad VELICIU

PROIECT DE DIPLOMĂ

System for Automatic Greenhouse

SPECIALIZAREA ELECTRONICĂ APLICATĂ

ÎN LIMBA ENGLEZĂ

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII
ȘI TEHNOLOGIA INFORMAȚIEI
UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

2024

Str. George Barițiu, nr. 26-28, 400027, Cluj-Napoca, tel: 0264-401224, tel/fax: 0264-591689, etti.utcluj.ro

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE ELECTRONICĂ
TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI
Specializarea: Electronică Aplicată (în limba engleză)

System for Automatic Greenhouse

Proiect de diplomă

Absolvent,
Vlad VELICIU

Decan,
Prof.dr.ing. Ovidiu POP

Președinte comisie,
Prof.dr.ing. Ovidiu POP

2024

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

FACULTATEA DE ELECTRONICĂ TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

Departamentul Electronică Aplicată

Titlul proiectului de diplomă:

System for Automatic Greenhouse

Descrierea temei:

This study proposes an automated greenhouse system with a Wi-Fi module, control components, sensors, and an integrated ATmega64A microcontroller on a PCB. Galvanic isolation is developed to prevent hazards from reaching delicate components. The ATmega64A communicates with ThingsBoard using the MQTT protocol, and water flow is determined using the Takagi-Sugeno fuzzy inference method based on soil humidity and air temperature. This solution offers a cost-effective, user-friendly control system and data logger, accessible anytime.

Locul de realizare:

la domiciliu

Data emiterii temei: 23.10.2023

Data predării temei: 05.07.2024

Absolvent,

Vlad VELICIU

Director departament,

Prof.dr.ing. Dorin PETREUS

Conducător,

Conf.dr.ing. Liviu VIMAN

Declaraţie pe proprie răspundere privind autenticitatea lucrării de diplomă

Subsemnata/ul **Vlad VELICIU**
legitimat cu CI seria SB nr. 889889 (conform copiei anexate prezentei declarații, copie semnată și certificată "conform cu originalul"), autorul lucrării

System for Automatic Greenhouse

elaborată în vederea susţinerii examenului de finalizare a studiilor de licență,
la Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Programul de studiu Electronică Aplicată (în limba engleză)
din cadrul Universității Tehnice din Cluj-Napoca
sesiunea iulie a anului universitar 2023-2024

declar pe proprie răspundere că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate în textul lucrării şi în bibliografie.

Declar că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor. Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de absolvire/licenţă/diplomă/disertaţie.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv anularea examenului de diplomă.

Sunt de acord ca, pe tot parcursul vieții, în cazul în care este necesar și se va dori verificarea autenticității lucrării mele să fiu identificat și verificat în baza datelor declarate de mine și conform copiei documentului de identitate menționat mai sus.

Data
04.07.2024

Nume și prenume
Vlad VELICIU

Semnătura

Absolvent: **Vlad VELICIU**
Conducător: **Conf.dr.ing. Liviu VIMAN**

SINTEZA PROIECTULUI DE DIPLOMĂ

System for Automatic Greenhouse

The development of an automated greenhouse system is suggested by this paper, along with a method for regulating and monitoring key parameters. It is comprised of a PCB-embedded ATmega64A microcontroller, five sensors, two control components, and a Wi-Fi module. Galvanic isolation was created to prevent risks on the lines outside from the board from getting to sensitive components. This isolation is thus designed between the components that have to be connected outside the 4-layer PCB and the ones soldered on it. Interchange of information between ATmega64A and ThingsBoard, an Internet of Things open-source platform, is accomplished by communication via the Wi-Fi module which uses the MQTT protocol. The Takagi-Sugeno fuzzy inference system used calculates the water flow depending on the air temperature and soil humidity. This solution resembles the ones already existing on the market, but it provides a cheap, easy-to-use implementation of a control system, as well as data logger through the fact that it is accessible anywhere anytime.

Sistem pentru seră automată

Această lucrare sugerează dezvoltarea unui sistem automat pentru o seră, precum şi o metodă de reglare şi monitorizare a parametrilor cheie. Sistemul este compus dintr-un microcontroller ATmega64A încorporat într-un PCB, cinci senzori, două componente de control şi un modul Wi-Fi. Izolarea galvanică a fost creată pentru preveni hazarduri de pe liniile din exteriorul plăcii să ajungă la componentele sensibile. Astfel, această izolare este proiectată între componentele care trebuie conectate în afara PCB-ului proiectat pe 4 layere şi cele lipite pe acesta. Schimbul de informaţii între ATmega64A şi ThingsBoard, o platformă Internet of Things open-source, se realizează prin modulul Wi-Fi care utilizează protocolul MQTT. Sistemul de inferenţă fuzzy Takagi-Sugeno utilizat calculează debitul de apă în funcţie de temperatura aerului şi umiditatea solului. Această soluţie seamănă cu cele deja existente pe piaţă, dar oferă o implementare ieftină şi uşor de utilizat a unui sistem de control, precum şi data logger prin faptul că este accesibil oriunde oricând.

Avizul conducătorului
Conducător,

Absolvent,

Content

1 Summary in Romanian

1.1 Descrierea temei

Schimbările climatice au avut un impact major asupra vieţii în ultimii ani, mai ales asupra agriculturii. Această problemă duce la afectarea lanţultui de alimentare la nivel global. O soluţie este reprezentată de implemntarea de sere care oferă un mediu controlat pentru dezvoltarea plantelor, acestea fiind o componentă de bază în asugurarea securităţii alimentare. Observând şi trendul de creştere al populaţiei, procentul de terenuri arabile ar trebui să crească în următorii ani, însă încălzirea globală afectează major acest aspect. O soluţie este reprezentată de sere, transformând terenurile în unele arabile prin controlul parametriilor într-un spaţiu închis. Cercetarea în domeniul dezvoltării serelor este un accelerată în ultimii ani, depăşind-o pe cea în domeniul agriculturii clasice. Aceasta se bazează pe îmbunătăţiri în ceea ce priveşte managementul consumului de energie pentru o seră cât şi pe uşurinţa utilizării unui sistem autonom de monitorizare şi control [1].

Majoritatea soluţiilor se bazează pe determinarea câtorva paramterii de bază, cum ar fi temperatura şi umiditatea din aer, umiditatea solului, intensitatea luminoasă şi concentraţia de gaze în aer, precum şi pe utililizarea de difereite tipuri de control pentru a dezvolta un sistem autonom. De asemenea, sistemele oferă posibilitatea de comunicare la distanţă prin wireless, folosind protocoale diverse (cum ar fi ZigBee) sau prin GSM (Global System for Mobile Communication). În ceea ce priveşte controlul parametrilor, sisteme care folosesc inteligenţă artificială sau reţele de senzori inteligenţi sunt implementate. Sistemul propus în această lucrare este similar cu aceste soluţii, monitorizând în plus nitorgenul, azotul, fosforul şi potasiumul din sol şi oferind controlul unei pompe de apă cu PWM (Pulse-Width Modulated) cu factor de umplere variabil. Determinarea factorului de umplere este realizată de un sistem de inferenţă fuzzy Takagi-Sugeno de ordin 0. Întregul sistem este implementat pe un PCB (printed circuit board).

1.2 Descrierea succintă a sistemului

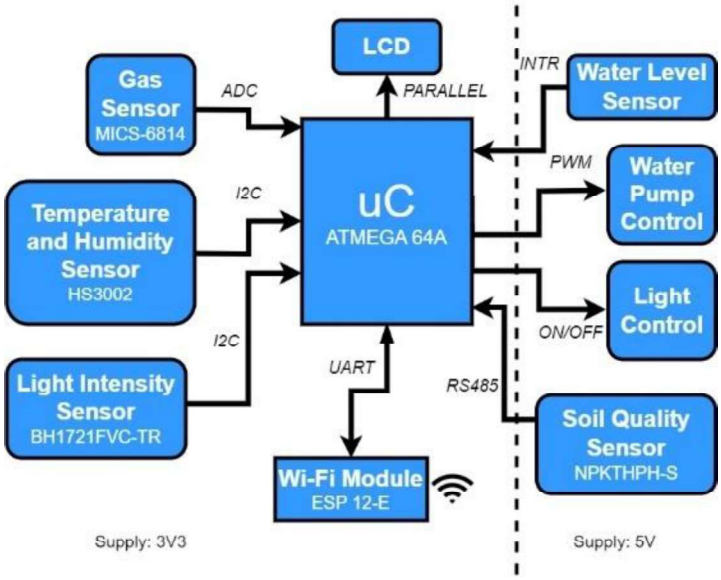

Figura 1.2.1. Schema bloc

Sistemul a cărui schemă bloc este prezentată în Figura 1.2.1., este compus dintr-un microcontroller ATmega64A care controlează un senzor pentru determinarea temperaturii şi

umidităţii aerului, HS3002, un senzor care măsoară intensitatea luminoasă, BH1721, unul pentru stabilirea concentraţiei a trei tipuri de gaze în aer (CO, NH_3, NO_2), MICS-6814, şi altul pentru calitatea solului. Comunicarea wireless este asigurată de modulul Wi-Fi, ESP-12E, care utilizează protocolul IEEE 802.11 b/g/n, transmiţând în banda de 2.4GHz spre o platformă IoT (Internet of Things). Pentru afişarea locală a parametrilor, un modul LCD (Liquid Cristal Display) a fost montat pe placă. Toate aceste componente sunt alimentate la o tensiune de 3.3V. Componentele care sunt conectate la distanţă faţă de placă sunt izolate galvanic faţă de cele de „semnal mic" care sunt mai scumpe şi greu de înlocuit astfel încât orice perturbaţie care poate apărea pe liniile de conexiune cu placa (de exemplu hazard) să nu poată să afecteze microcontrollerul sau modulul wireless. Aceste componente sunt releul care comandă circuitul pentru controlul luminii, circuitul pentru comanda pompei de apă, cel pentru senzorul de calitatea solului şi cel pentru senzorul de nivel de apă. Alimentarea acestora se face la o tensiune de 5V.

1.3 Design-ul sistemului şi programarea

Microcontrollerul foloseşte protocolul I2C (Inter-Integrated Circuit) pentru comunicarea cu senzorii HS3002 şi BH1721. Pentru funcţionarea acestui protocol rezistenţe de pull-up sunt plasate pe liniile SDA şi SCL. Protocolul este unul care permite comunicarea într-un singur sens pe cele două linii şi constă în stabilirea conexiunii prin adresarea de către microcontroller a senzorului, adică transmiterea adresei acestuia (care e formată din 7 biţi) la care se adaugă un bit de „write" („0"). Pentru semnalizarea faptului că se doreşte citirea valorii măsurate, se va transmite adresa la care se adugă un bit de „read" („1"), iar senzorul va transmite datele. După fiecare pachet de 8 biţi (un octet) transmis pe bus, un bit de acknowledge („0") va fi emis de către dispozitivul căruia i se adresează pachetul respectiv. Un bit de start („0") este transmis de către microcontroller pe bus de fiecare dată când o astfel de conexiune se doreşte a fi stabilită şi unul de stop („0") pentru oprirea conexiunii. La HS3002 4 octeţi de date sunt transmişi, primii doi conţinând valoarea umidităţii aerului, iar următorii doi reprezintă valoare temperaturii aerului. În cazul senzourului BH1721, valoare intensităţii luminoase măsurate este reprezentată pe doi octeţi.

Senzorul pentru determinarea calităţii solului foloseşte protocolul RS-485. Acesta este unul diferenţial pentru care comunicarea cu microcontrollerul nu se poate realiza direct. Din acest motiv este nevoie de un transceiver, MAX485, care asigură conversia UART (Universal Asynchronous Receiver-Transmitter) - RS-485. Impedanţa caracteristică a liniilor diferenţiale pentru acest protocol este 120Ω. Ştiind că diferenţa de tensiune minimă (între liniile diferenţiale) care poate fi detectată la recepţie este de 200mV (diferenţa pozitivă indică „1", iar cea negativă „0"), o parte din domeniul de tensiuni rămâne nedefinită. Astfel se dimensionează o configuraţie care determină ca şi domeniul de tensiune între 0V şi 200mV să fie interpretat ca „1". Pentru aceasta, o rezistanţă de valoarea 120Ω este plasată între cele două linii diferenţiale, cât şi rezistenţe de pull-up şi pull-down care asigură o diferenţă de tensiune de cel puţin 200mV, chiar dacă senzorul transmite o diferenţă mai mică. Doar pentru diferenţe de tensiuni în domeniul de la 0V la 200mV se poate realiza această „shift-are" în tensiune pentru posibilitatea interpretării ca „1".

Senzorul pentru detectarea concentraţiei de gaze din aer, MICS-6814, este format din semiconductori care au propietatea de a-şi schimba rezistenţa odată cu variaţia concentraţiei. Senzorul are un strat de încălzire şi unul sensibil pentru fiecare tip de gaz. Foaia de catalog a acestui senzor prezintă pentru fiecare din cele trei tipuri de gaze doar dependenţa R_s/R_0 de concentraţie (exprimată în scară logaritmică), unde R_s este valoarea rezistenţei măsurate şi R_0 reprezintă valoarea rezistenţei în cazul măsurătorii în aer curat. În consecinţă, valoarea R_0 trebuie determinată şi liniarizarea pe intervale este necesară pentru a putea implementa ecuaţiile în microcontroller, ocupând memoria într-un mod mai eficient faţă de scrierea în memorie a unor ecuaţii exponenţiale. În serie cu stratul sensibil se conectează o rezistenţă de valoare 56kΩ astfel încât curentul să poată fi controlat pentru a nu distruge acest strat. Pentru funcţionarea senzorului este necesară trecerea unui curent prin stratul de încălzire care este asigurată prin pornirea din microcontroller a unui

tranzistor pentru o perioadă de minim 30s după care procesul de încălzire este oprit. Curentul necesar acestui proces fiind de 88mA, încălzirea continuă va consuma un curent considerabil care nu este necesar, schimbarea concentraţiei gazelor neîntâmplându-se instant. Din acest motiv, încălzirea este oprită pentru o perioadă de 150-180s înainte de a fi pornită din nou. Procesul de determinare a rezistenţei este realizat prin colectarea de eşantioane folosind convertorul analog-digital al microcontrollerului. Astfel, un timer este pornit şi la fiecare 66ms un eşantion din convertor este citit, un total de 20 de eşantioane fiind culese. Pentru aceasta se foloseşte Timer-ul 2 care este pornit după ce perioada de încălzire se termină şi este oprit după ce s-au cules toate cele 20 de eşantioane pentru toate cele 3 tipuri de gaze. Se realizează apoi media aritmetică a eşantionelor. Pentru determinarea rezistenţei se determină curentul care trece prin acest strat sensibil (prin măsurarea tensiunii pe rezistenţa de 56kΩ şi împărţirea rezultatului la valoarea acestei rezistenţe); apoi se determină tensiunea pe stratul sensibil şi prin împărţirea acesteia la valoarea curentului aflată anterior, se poate stabili valoarea R_s. Experimental s-a observat că rezistenţa în aer curat R_o pentru CO este de 2kΩ care corespunde unei concentraţii de 8ppm şi pentru NO_2 a fost măsurată o rezistenţă de 50.25kΩ care semnifică o concentraţie de 0.22ppm.

Senzorul de nivel este unul analogic, având la ieşire o tensiune de 2.5V când rezervorul de apă este plin. Acest sensor este conectat în partea izolată galvanic de microcontroller, iar optocuplorul care este folosit pentru realizarea conexiunii optice dintre cele două zone, nu poate transmite tensiuni analogice. Din acest motiv a fost proiectat un comparator cu histereză în configuraţie neinversoare care are la ieşire doar două nivele: 0V şi 5V. Pragurile pentru comparator au fost stabilite experimental la valorile 2.1V şi 2.3V, astfel încât la fiecare schimbare a stării rezervorului de apă să se activeze întreruperea externă pentru microcontroller. Pompa de apă va fi oprită automat când este detectat front descrescător (rezervorul este gol).

Pentru optocuploare un curent de 2.1mA este proiectat prin partea unde este LED-ul ceea ce determină un curent de colector în partea unde este tranzistorul de 1.5mA. Pentru a îmbunătăţi rejecţia de mod comun, rezistenţe care limitează curentul de valori egale sunt plasate simetric faţă de pinii LED-ului.

Pompa de apă este comandată de un tranzistor care asigură curentul necesar. Acesta este comandat cu un semnal PWM generat de microcontroller folosind Timer-ul 0 pentru a comuta starea pinului pe care se generează semnalul. Factorul de umplere este determinat de sistemul de inferenţă fuzzy, ţinând cont de temperatura aerurlui şi de umiditatea solului. Pentru fiecare dintre aceşti parametrii s-au definit 3 mulţimi fuzzy, „Small”, „Medium”, „High” pentru temperatură şi „Dry”, „Fine”, „Wet” pentru umiditate. În ambele cazuri primele două mulţimi sunt triunghiulare, iar ultima este trapezoidală, formându-se o partiţie fuzzy (suma gradelor de aparteneţă a unei valori la totate cele 3 mulţimi fiind 1). La ieşire sunt 4 mulţimi single tone (gradul de aparteneţă este 1 doar la cele 4 valori, în rest fiind 0) care reprezintă valorile factorului de umplere (0%, 25%, 50%, 100%). S-au definit 9 reguli care acoperă toate combinaţiile posibile dintre mulţimile fuzzy de intrare, de exemlpu „Dacă temperatura e Medium şi umiditatea e Fine factorul de umplere va fi 25%”. Se determină gradul de apartenenţă a temperaturii măsurate la mulţimea „Medium”, a umidităţii citite la mulţimea „Fine”, iar apoi se utilizează operatorul produs (pentru „şi”) între cele două grade de aparteneţă pentru a afla gradul de activare a acestei reguli. Identic se procedează pentru fiecare din cele 9 reguli, apoi se realizează media ponderată între gradele de activare şi valorile factorului de umplere din mulţimea de ieşire pentru determinarea rezultatului final. Acest ultim proces se numeşte defuzzyficare. Temperatura precum şi umiditatea dorite pot fi controlate de către utilizator prin interfaţa de la platforma ThingsBoard. Aceste valori reprezintă vârful mulţimilor triunghiulare „Medium”, respectiv „Fine”, lăţimea acestor mulţimi rămânând nemodificată în cazul schimbării valorilor dorite, acestea fiind doar shiftate în domeniul de valori. Pentru funcţionarea optimă a pompei de apă factorul de umplere determinat trebuie să fie peste 20%, toate valorile rezultate sub acest prag vor corespunde cu factor de umplere 0, deci pompa va fi oprită. Un tranzistor este folosit pentru a controla pompa, asigurând curentul necesar. Astfel prin optocuplor, în partea unde este LED-ul, se va proiecta un curent de 4mA, determinând un curent în baza acestui tranzistor de 5mA, rezultând un curent de aproximativ 300mA pentru pompă.

Controlul unui circuit conectat extern care comandă iluminatul în seră se face printr-un releu. Utilizatorul poate porni şi opri releul printr-un buton din interfaţa dedicată.

Modulul Wi-Fi realizează comunicarea cu platforma ThingsBoard, fiind una open-source. Conexiunea se realizează prin conectarea cu numele de host „demo.thingsboard.io" la portul specific protocoului MQTT (Message Queuing Telemetry Transport) utilizat, 1883, protocol de strat Aplicaţie în stiva TCP/IP. Acest protocol se bazează pe structura publisher/subscriber între două entităţi (în aces caz modulul şi serverul de la platformă), sensul de comunicare fiind controlat de un broker (platforma în acest caz), modulul şi serverul interschimbându-şi rolurile pe parcursul funcţionării. Conexiunea cu dispozitivul în cadrul platformei se realizează prin subscripţia cu token-ul specific, efectuându-se şi abonarea la RPC (Remote Procedure Call) request-uri prin care modulul primeşte date atunci când utilizatorul modifică elemente de control în interfaţă (butoane, switch-uri). Datele sunt formatate având un „topic" în care este specificat tipul mesajului, request sau response şi în şiruri de tip JSON (Javascript Object Notation), „message", care conţin denumirea parametrului măsurat şi valoarea acestuia. Astfel, când se realizează transmisia de date de la modul la platformă, modulul efectuează transmiterea topicului (unde sunt publicate) şi serializarea acestora (conversia în format JSON), iar la primirea datelor de la platformă se face deserializarea. Transmisia de date de la modul se face continuu, actualizând astfel parametrii măsuraţi. Consumul de curent fiind însă unul semnificativ (poate ajunge la 170mA), publicarea de date către platformă se face odată la 5 iteraţii de măsurare efectuate de ATmega64. De asemenea, pentru a asigura o gestionare optimă a consumului de curent atunci când se realizează transmisia, un condensator electrolitic este plasat în vecinătatea modulului şi a circuitelor adiacente, acesta având rolul de rezervor local de energie astfel încât o cerere mare de curent venită de la modul să poată fi rezolvată într-un timp cât mai scurt. Transmisia de date între ATmega64A şi ESP-12E este realizată folosind UART. Pentru aceasta, biblioteca SoftwareSerial este folosită cu rata de baud de 9600. Informaţia transmisă de la ATmega64A la ESP-12E este împărţită în numere până la 100 (sau parte întreagă şi parte fracţionară ca să poată fi trimisă pe bus-ul de 8 biţi), iar apoi este reconstituită la recepţie. Transmisia de la modul la ATmega64 se realizează prin întreruperea de recepţie a portului serial. Astfel, când utilizatorul execută o schimbare a parameteriilor de control (butonul AUTO care cuplează/decuplează sistemul de control fuzzy, butonul pentru controlul releului, butonul pentru controlul pompei de apă – activ doar când sistemul de control este decuplat şi duce la pornirea pompei cu factor de umplere de 50% şi reglajul pentru temperatura şi umiditatea dorite), un octet este transmis care activează întreruperea de recepţie a portului serial şi care reflectă controlul făcut de utilizator în interfaţă. Programarea ESP-12E este realizată folosind mediul Arduino, prin portul serial.

Pentru funcţionarea microcontrollerului ATmega64A, un oscilator extern de frecvenţă 8MHz este conectat, dispozitivul fiind alimentat la o tensiune de 3.3V, frecvenţa a fost aleasă pentru a păstra punctul de funcţionare în interiorul ariei sigure de operare. Biţii de fuse trebuie reglaţi şi ei pentru a specifica microcontrollerului că semnalul de ceas trebuie preluat de la pinii unde este conectat oscilatorul. Programarea se face prin intermediul ICSP (In-Circuit Serial Programming), utilizând driverul usbasp şi utilitarul avrdude pentru comunicarea modulului de programare cu PC-ul. Astfel programarea se va face direct din linia de comandă folosind fişierul hex generat de Microchip Studio (mediul de programare folosit).

Alimentarea celor două zone este realizată cu baterii Li-Ion de 7.4V şi 6.8Ah, iar tensiunile necesare zonelor sunt obţinute prin regulatoare de tensiune.

1.4 Design-ul PCB-ului

PCB-ul dezvoltat este pe 4 straturi distribuite astfel: layer-ele top şi bottom sunt folosite pentru rutarea traseelor de semnal şi au grosimea cuprului de 35µm, al doilea layer este folosit pentru planul de masă, iar al treilea pentru cel de putere (alimentare). Pe layer-ele interne grosimea cuprului este de 17.5µm. Layer-ele sunt prezentate în Figura 1.4.1. Acest stack-up este unul

standard de la producător, JLCPCB. Lăţimea traseelor de semnal este de 0.3mm, asigurând o impedanţă caracteristică de 50Ω, iar pentru traseele de putere lăţimea aleasă este de 0.4mm pentru a asigura o capabilitate în curent mai mare, de până la 1.9A. Via-urile pentru trecerea de pe un layer pe altul au diamterul găurii de 0.3mm şi cel al pad-ului de 0.7mm. Astfel se asigură atât capabilitate în curent suficientă, cât şi inductanţă şi capacitate parazită mică. Traseele diferenţiale pentru RS-485 sunt rutate cu o lăţime de 0.12mm şi o spaţiere între trasee de 0.2mm pentru a asigura impedanţa caracteristică a liniei de 120Ω.

Traseele rutate pentru I2C şi cele de conexiune ale oscilatorului cu microcontrollerul sunt gardate cu via-uri către planul de masă pentru a diminua influenţa acestor trasee asupra celor adiacente, semnalele care trec prin aceste trasee fiind de frecvenţe mai mari faţă de restul de pe placă (frecvenţa I2C de 100kHz şi cea a oscilatorului de 8MHz, deci în jurul acestora se creează un câmp electromagnetic semnificativ, putând induce perturbaţii pe traseele apropiate). De asemenea, sub antena de pe modulul Wi-Fi s-a realizat o porţiune de cut-out în placă pentru a reduce influenţa câmpului emis de 2.4GHz asupra traseelor de pe placă astfel încât informaţia purtată de acestea să nu fie eronată (să nu fie afectat timpul de creştere sau durata unui bit). Tot din motive de integritate a semnalelor s-a realizat un plan de masă pe layer-ele externe care curpinde suprafaţa dintre trasee şi pad-uri pentru ca perturbaţiile să fie drenate spre masă şi să nu afecteze traseele. Pe marginea planurilor sunt plasate via-uri pentru a preveni fenomenul de ground bouncing, deci pentru a asigura acelaşi potenţial pe întreaga suprafaţă a planului, acesta având o extindere semnificativă.

Din punct de vedere al curentului consumat, zona alimentată la 3.3V are un consum maxim de 850mA, iar cea alimentată la 5V, 450mA. Astfel, pe regulatorul de 3.3V este disipată o putere de 3.48W care este una semnificativă pentru acest dispozitiv. Din acest motiv este necesară proiectarea pentru a crea o a doua cale de disipare termică prin partea de jos a plăcii, reducând astfel temperatura maximă care poate apărea pe joncţiune. Plasând 9 via-uri sub dispozitiv, se reduce rezistenţa echivalentă joncţiune-ambiant, rezultând într-o temperatură maximă pe joncţiune de 106°C, mai mică decât cea maximă suportată de 125°C. Pentru regulatorul de 5V s-a procedat la fel, plasându-se via-uri, însă în acest caz disiparea puterii nu duce la o creştere semnificativă a temperaturii maxime pe joncţiune (doar 52.1°C).

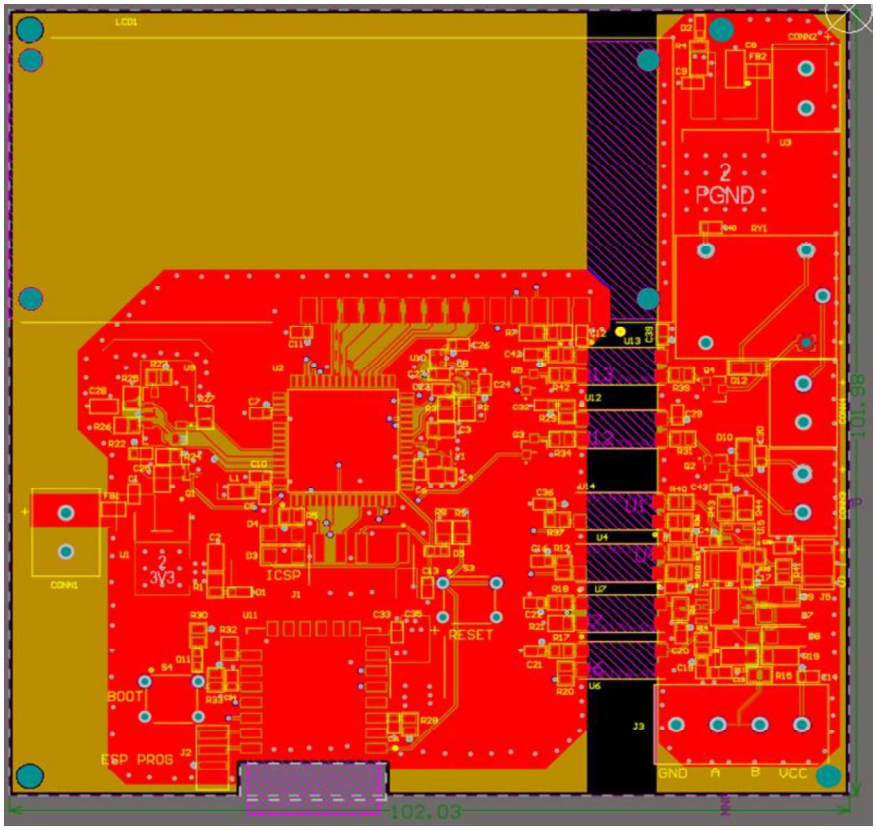

Figura 1.4.1. Layer-ele

1.5 Concluzii

Sistemul este aşadar unul uşor de implementat la un cost relativ mic, asigurând monitorizarea şi controlul permanent al parametrilor prin comunicarea wireless. O îmbunătăţire este reprezentată de alimentarea asigurată de panouri solare în locul folosirii bateriilor care se pot reîncărca. Pentru aceasta se poate folosi un convertor flyback astfel încât izolarea galvanică a celor două zone de pe placă să se păstreze (să nu aibă punct comun de masă). Pentru a putea conecta panourile la convertorul flyback, un invertor este necesar pentru a transforma curentul continuu produs de panouri în curent alternativ, alimentând intrarea convertorului. De asemenea, se pot automatiza mai mulţi parametrii, numărând cu un timer perioada în care intensitatea luminoasă este peste un anumit prag, asigurarând 16 ore de lumină (perioda ideală de lumină pentru o plantă) într-o zi.

2 Work planning

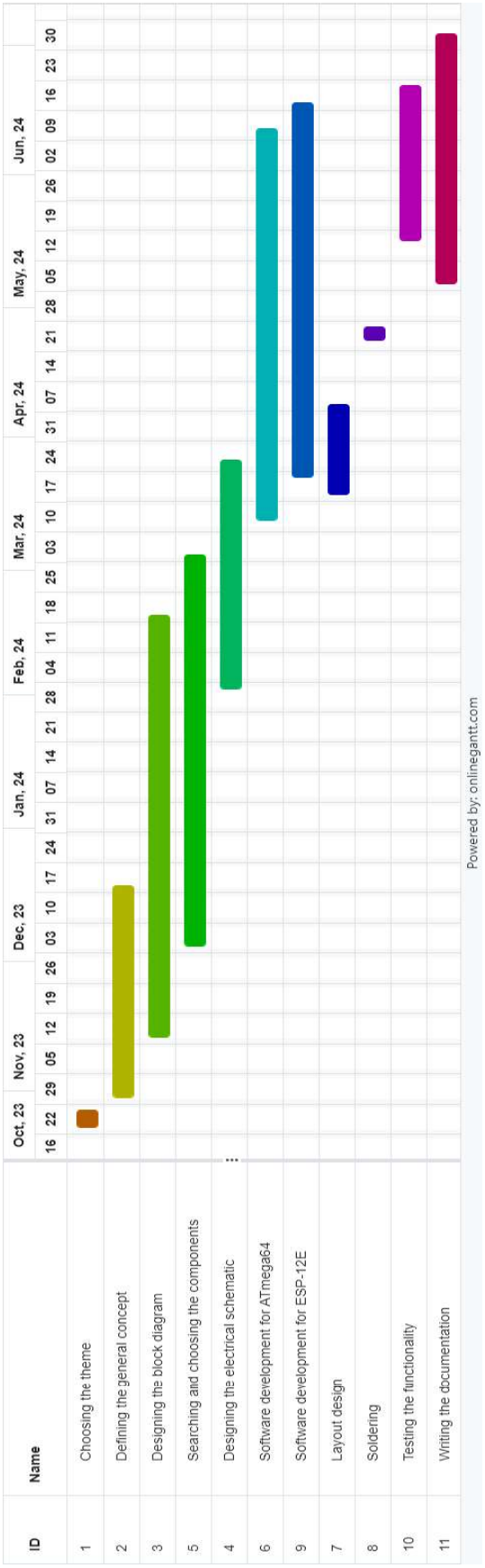

Figure 2.1. Gantt diagram for work planning

3 State of the Art

In recent years, the increasingly climate changes have visibly impacted our lives, particularly in agriculture, thereby affecting global supply chains. Greenhouses represent a solution to mitigate the effects of polluted air and soil on plant health, plants being the fundamental components of the supply chain. By providing a controlled environment, greenhouses enable precise monitoring and management of air and soil conditions, resulting in optimal plant growth and supporting supply chain efficiency. Also, there has been a higher demand for food in the last years as a consequence of the increasing population worldwide, making the supply of food one of the biggest challenges of the current century. Only less than a quarter of the total farmed land has automatic irrigation systems, considering that the needed farmed land needs to increase by 1.3% every year. Thus, a problem regarding the availability of the farmland is raised. Not only is there a necessity to convert more land into farmland but also to make this conversion in a sustainable way in order to influence the environment as little as possible. This is possible only by developing greenhouses in a sustainable manner [1]. A comparison of the trends in greenhouse technology and agriculture research over the past years is presented in Figure 3.1. An increase in the research in greenhouse development can be observed, signaling the importance of this subject for the future.

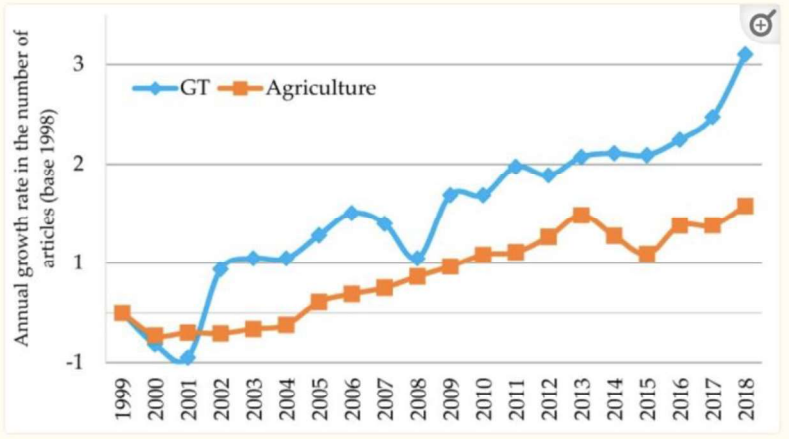

Figure 3.1. Trends in greenhouse technology (GT) and agriculture research [1]

Greenhouses have evolved together with technology, as technology has gained a huge impact on our lives lately. In the last years, it has shown its contribution to facilitate an easy use of a greenhouse through the development of dedicated systems for monitoring and control. The main features of such a system are related to the irrigation, design of an optimum structure, ensuring best conditions for the soil, achieving best power consumption of the whole system, control of the climate parameters and identifying diseases [1]. Nowadays, food, energy and financial security, are the three major fundamentals for life. Food influences the health diet of people which is becoming more and more necessary as there can be seen that the number of diseases caused by improper nutrition have alarmingly grown in the last years. Energy consumption of every developed system needs to be kept lower and lower as the renewable energy sources that replace the conventional ones for a cleaner environment, cannot provide a comparable amount of energy as the old ones yet. Financially there can be stated that the trend is to develop a system or a product which needs a very low budget for its construction [2].

Therefore, for the development of a greenhouse, a “smart” system needs to be built. For instance, artificial light can be used to complement natural light whose intensity can vary. In this way, proper growth of the plant can be ensured. There have been also developed some facilities where plants get only artificial light so that a cycle of 16 hours of light and 8 hours of darkness can be fully controlled in an automatic manner. Automating the control of the vital parameters for a

greenhouse, the human intervention in this process is reduced to minimum, to an observer of the whole [2].

Hence, many systems for greenhouses were developed to fulfil the above requirements and trends that are impacting our lives nowadays.

3.1 Comparison between solutions

Several solutions have been already explored, many of them using an ARM based microprocessor and embedded web server to facilitate the real-time transfer of information between the system and the user. Such a system is composed of several sensors, that are seen as nodes by the server as they monitor different areas, Analog-to-Digital converters, and other circuit elements. The Wi-Fi protocol used is ZigBee in the 2.4GHz band, a platform from Samsung where are the data is stored is also made use of. Communication based on HTTP requests is achieved and a Boa web server is used, a single task web server, handling multiple connections by a list of HTTP requests, instead of establishing a special connection for each request, saving thus resources [3].

Another solution is developed around a PLC (Programmable Logic Controller) and a SCADA (Supervisory Control and Data Acquisition) system to build an automatic system. The sensors used form a WSN (Wireless Sensor Networks) that sends data to the control unit and this data is then processed using AI (Artificial Intelligence) algorithms to improve the use of water and energy [4].

A different way to achieve the control of some parameters in a greenhouse is by developing a fuzzy inference system, the parameters having a nonlinear relationship and a classic control system is more difficult to implement. A prediction and control of the humidity is made based on ventilation, heating, artificial light, CO2, external temperature, and others. This is implemented with a Mamdani inference system and the clustering technique, as it ensures a better premise for setting up the rules for the system [5].

Another approach implements a system that monitors several parameters, such as temperature (with DHT11 sensor), soil moisture, soil pH and light intensity. SMS (Short Message Service) is sent via a GSM (Global System for mobile communication) module to transmit the values measured. Also, the user can control a cooling fan, water pump, light, and exhaust fan by sending SMS. The parameters are also sent to a database, which is accessible on a mobile phone, through Ethernet. An Arduino board manages all these actions [6].

Consequently, there are many solutions that approach a greenhouse implementation, most of them monitoring five essential factors: air temperature, air humidity, soil moisture, light intensity and CO2 level [7]. The solution that is proposed by this paper resembles the ones presented, but more soil parameters and gas concentrations will be monitored, and a control for water pump and circuit for turning on and off a light source will be implemented.

4 Theoretical Fundamentals

For measuring each of the parameters mentioned above, specific sensors are needed. These are managed by a microcontroller that also controls a water pump and a relay for switching on and off a circuit for light control, as well as Wi-Fi communication. The whole system is implemented on a PCB (Printed Circuit Board). All these elements will be theoretically analyzed in this section.

4.1 Temperature / Humidity sensor and Light Intensity sensor

Temperature is the most important factor in the growth of plants. For appropriate growth, different types of plants require specific temperatures, so this parameter needs to be properly monitored. Humidity is also essential, as plants expel warm and moist oxygen when they grow but a high relative humidity in a closed space, like the one from a greenhouse, can lead to the development of diseases. Light is another important factor, as plants use it for photosynthesis. Controlling the light intensity in a greenhouse would also allow to keep the plants safe from UV radiation and could ensure a symmetric distribution of the light [7].

4.1.1 Types of temperature sensors

A. Thermocouples

These sensors have a wide temperature range (-180°C - 2320°C) and quick response times. They are made by joining two dissimilar conducting metal wires, causing a Seebeck effect (a temperature difference of two different type of wires causes a voltage difference that is used to calculate the temperature). A disadvantage is the small output voltage which can cause problems in measuring the temperature. Precise amplification is required and external noise over the wires can affect the result. Cold junction appears between the thermocouple's wires and the wires of the external circuit that causes another Seebeck effect that must be compensated. There are integrated solutions (with precise ADC – analog to digital converter, low noise, and compensation of the cold junction) [8].

B. Resistance temperature detector

This sensor is based on the increasing resistance as temperature increases and the resistance vs temperature relationship is known for any material. Platinum is the most used material for manufacturing such a sensor because it offers an almost linear response to temperature changes, is stable and accurate and offers a wide range. This sensor is suitable for low ranges (-200°C - 500°C). There are two, three or four wires configurations available, depending on the accuracy needed [8].

C. Thermistors

This type of sensor is like the previous type. It has a much stronger dependency, a change in temperature causing a change in resistance that is measurable. The most used thermistors are the ones whose resistance increases when temperature decays, so they have negative temperature coefficient. They are cheaper but less accurate than resistance temperature detectors and have a non-linear resistance vs temperature characteristic, as shown in Figure 4.1.1. It is commonly used in a voltage divider configuration with an ADC [8].

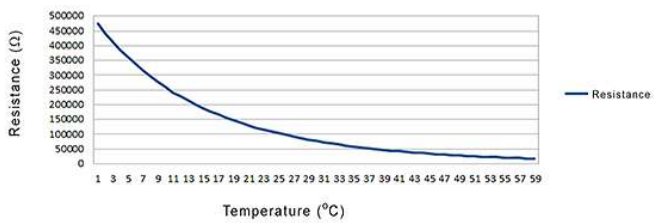

Figure 4.1.1. Resistance vs Temperature characteristics for a thermistor [8]

D. Semiconductor based ICs

This sensor works with dual ICs. They contain two similar diodes or transistors with temperature-sensitive voltage and current characteristics to measure the temperature changes effectively. They give a linear output, are sensitive, but are less accurate at 1°C-5°C and have slow responsiveness (5s - 60s). Range is between -70°C - 150°C [8].

4.1.2 Types of humidity sensors

A. Capacitive

This sensor measures the relative humidity. A thin strip of hygroscopic polymer film is placed between two electrodes (conductive plates), as illustrated in Figure 4.1.3. This material is most used in MEMS (Micro Electromechanical Systems) humidity sensors. This technology refers to a fixed part and a "moving" one, between them several structures that resemble capacitors are formed. Thus, these capacitances change, and this change needs to be measured [11]. The capacity of the metal oxide changes with the atmosphere's relative humidity, this material being hygroscopic. The capacitance will change because of the dependence of the dielectric, changing its permittivity, to the relative humidity (a function of the ambient temperature and the water vapors pressure) [9]. The equation (4.1.1) depicts this dependency. To measure the capacitance, an LC-oscillator can be built, whose frequency varies with the relative humidity [10]. These sensors can have the best response time and sensitivity among the humidity sensors if the sensing material thickness is reduced to minimum, diminishing thus the parasitic capacitance [11].

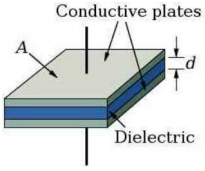

Figure 4.1.3. Structure of a capacitor [9]

$$C = \varepsilon * \frac{A}{d};\ \varepsilon = dielectric\ constant \tag{4.1.1}$$

B. Resistive

This type of sensor is made up of materials with relatively low resistivity and the resistivity varies with humidity in an inverse proportional way. The low resistivity material is placed on top of the two electrodes. The contact area between the electrodes and the material has to be increased as much as possible and the resistivity of the material changes when the top layer absorbs water, thus the technology used is MEMS. Commonly used materials are salt, conductive polymers [10].

C. Thermal conductivity

This type of sensor is used to measure the absolute humidity. It measures the thermal conductivity of dry air and air with water vapors, and the difference is called to as absolute humidity. Two thermistors with negative temperature coefficients are used. One is isolated in a room filled with dry Nitrogen and the other one is exposed to open air through small venting holes. When powering on the circuit, the resistance of the two are measured and the difference is direct proportional to the absolute humidity [10].

4.1.3 Types of light sensors

The working principle of these sensors is based on the photoelectric effect. In an atom, the energy band occupied normally by electrons is the valence band. In a conductor, this band is typically half filled, allowing the electrons to move to the conduction band, generating thus a current and leaving an electron hole in the valence band, while in an insulator this band is

completely filled, not allowing electrons to get excited to the conduction band. In a semiconductor material, the valence band is generally filled but the energy needed for an electron to overcome the bandgap (space between the valence and conduction band), is much smaller than in an insulator. This energy needed for the electrons to get excited to the conduction band is typically between 1.12eV-1.42eV, depending on the material. This amount corresponds to the one carried by a photon of infrared and visible light, so that if it gets in contact with an electron, it has the capability to excite the electron to the conduction band, generating a current. There are two operating modes: photovoltaic effect when a voltage is generated by the freed electrons and photoconductive when the photosensitive element allows current to pass [12]. In what follows, several types of light sensors are presented.

A. Photodiodes

These types of sensors convert light to electrical current and consist of a p-n junction [13]. Figure 4.1.5. depicts the transfer characteristics, where the photoconductive mode in the third quadrat occurs and the photovoltaic mode in the fourth one. P_i represents the different light intensities. To use the photodiode as a light sensor, not as a current generator, the functioning point of the device needs to be set in the third quadrant [14].

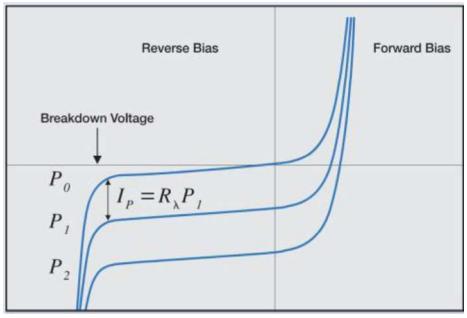

Figure 4.1.5. V-I characteristics of a photodiode [14]

A typical circuit configuration is depicted in Figure 4.1.6. The photodiode is in photoconductive mode, generating when light is sensed a current flowing from the cathode to the anode and through the negative feedback loop of the operational amplifier A_1, respectively through R_F. At the output of A_1 the voltage on the resistance R_F is present, as the inverting and non-inverting inputs of A_1 are at 0V. This voltage is then amplified through A_2. The capacitor C_F has the role to increase the stability of the system, introducing a pole in the transfer characteristics.

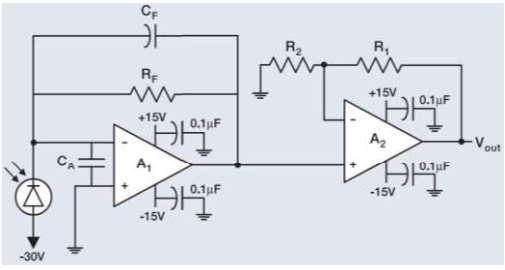

Figure 4.1.6. Typical circuit configuration [14]

B. Photoresistors

These resistors have the characteristic that their resistance decreases with the increase of the incoming light, illustrating the principle of the photoelectric effect [13].

C. Phototransistors

They are similar to regular transistors, amplifying or switching at incident light in the base [13].

To measure the air temperature and humidity, a single sensor is chosen that is based on the MEMS technology. For detecting light intensity, an integrated sensor based on a photodiode was

found most suitable. Both sensors are digital and communicate using the I2C (Inter-Integrated Circuit) protocol.

4.1.4 The I2C protocol

This protocol is a two-wire serial one, composed of a serial data line (SDA) and a serial clock line (SCL), allowing communication to distances up to one meter. The communication uses packets of bytes that are uniquely addressed for each device, and it is a half-duplex, meaning that only one device (master or target) can transmit data over the bus. The SCL line is controlled by the master, and it is used to synchronize the data. SDA is the line where the data bytes are transmitted. Only the master can start and stop communication over the bus, removing the problem of contention. The communication is initiated by the master by sending the corresponding address of the target device over the bus and thus the device will know that the future data is intended to reach it or that it has to send data to the master. To unbound this, the master needs to stop the communication [15].

Generally, a device that uses I2C protocol has its SDA and SCL pins in open-drain configuration, such that external pull-up resistors to the common voltage supply are needed for proper functioning of the bus. As Figure 4.1.7. (left) illustrates, the idle state of the lines (when the transistor is off) is logic "1" because of the pull-up resistors. A transition from HIGH to LOW state means the transistor turns on and this fall time is typically a small one, depending only on the drive strength and any bus capacitance on the line. Turning off the transistor means a transition from LOW to HIGH, the rise time being slower because the line is pulled up against the bus capacitance. The rise time depends on the value of the pull-up resistor, a lower resistance resulting in a faster transition, but it requires more power and a higher one results in a slower transition with less power consumption. There are also push-pull configurations where the external pull-up resistor is replaced with an internal PMOS transistor. The problem with this configuration appears when a device pulls its output HIGH and the other one pulls it LOW, such that the line gets in an undefined state, as Figure 4.1.7. (right) exemplifies [15].

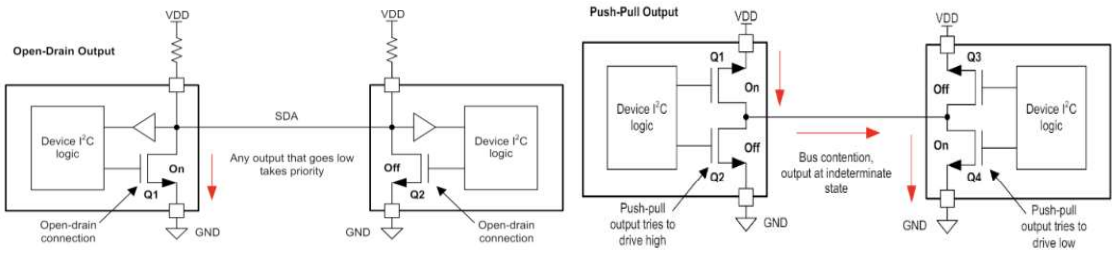

Figure 4.1.7. Structure of an SDA/SCL pin and the connection to another device [15]

Figure 4.1.8. shows the structure of an I2C communication frame. Thus, to initiate the communication a logic "0" start bit needs to be sent, followed by the 7-bit address of the target device (implying that there can be 2^7-1 unique addresses, so devices that are interconnected in daisy chain configuration), appending an 8^{th} bit that specifies the action made (read - data from device / write – addressing the device). An Acknowledge bit (ACK) is sent after every successful reception of a packet of bytes by the device to which the data is addressed. The data on the SDA bus is available before the rising edge of the clock from the SCL line, such that when the rising edge occurs, the bit is fetched. To end the communication, a STOP bit, active LOW, is sent [15].

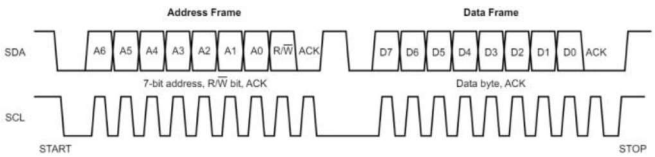

Figure 4.1.8. I2C communication frame [15]

To proper dimension the pull-up resistors, equation (4.1.2) [16] for a frequency of the serial clock f_{SCL} < 100kHz and (4.1.3) for f_{SCL} > 100kHz. 3mA refers to the maximum current through the structure composed by the pull-up resistor and the NMOS transistor, 0.4V is the typical voltage that drops on the transistor, the times at the numerator from the right-hand fraction represents the typical rise and fall times and C_b is the capacitance of the bus [16].

$$\frac{V_{CC}-0.4V}{3mA} < R_p < \frac{1000ns}{C_b}; C_{b\_max} = 400pF \quad (4.1.2)$$

$$\frac{V_{CC}-0.4V}{3mA} < R_p < \frac{300ns}{C_b}; C_{b\_max} = 400pF \quad (4.1.3)$$

4.2 Soil Quality sensor

Among all the soil parameters, the moisture is the most important. The wrong amount of water in the soil can damage the plant, leading to poor yielding, root disease or wastage of water. Water ensures that the plant receives the right amount of nutrients [7]. The soil PH is also an important factor. The PH makes available or removes certain ions from the soil. High acidity (PH much below 7) in soil means that it has toxic amounts of aluminum and manganese and that the soil organism can move with difficulty. In areas with a high number of rainfalls the PH tends to get acid, while in the ones with less rain it is alkaline (PH around 8.5-10). Most agricultural plants require a PH of 6.5 in mineral soils and 5.5 in organic ones. Soil temperature also determines the biological and chemical processes in the soil. For instance, for planting a certain soil temperature is needed so that the roots of the plants don't get damaged. It also regulates the germination, root growth and the availability of nutrients, as well as the morphological character of the root because a high amount of water at a low temperature of the soil can affect the chemical processes within the root. Nutrients are also helpful in the growth of the plant, its reproduction and represent a measure of the soil's fertility. Some important nutrients are nitrogen, phosphorus, potassium, calcium, magnesium and many others [17].

4.2.1 Functioning of the soil quality sensor

The soil moisture measuring is performed with the help of stainless-steel probes so that these probes can be buried in the soil and stay there for a long time without affecting the measurement result. There are three principles for performing this: *capacitive*, meaning that it is composed of a capacitor that changes its capacitance with the moisture level (similar to the air humidity capacitive sensor). The capacitance value is than converted to an electrical measurable signal (with the help of an LC oscillator). The *resistive* based sensor consists of a resistor that is moisture-sensitive, and it's made of materials that have good water absorption capabilities, determining a change in resistance that is then measured. The *ion-sensitive* principle is characterized by a field-effect transistor that contains ion-based solution in its gate. This membrane that is formed can sense the concentration of ions and then, with the help of a transducer, this information can be converted to an electrical signal.

The soil temperature sensor is like the air temperature one, meaning that it consists of a thermistor (sensing a change in resistance with the temperature). The difference is that soil temperature sensors are built to last longer.

A soil PH sensor is made up of a metal sensor that is a direct link with the soil and uses an oxide reaction to generate current. This current will drive the PH value in the functional-switching device and then this value can be used.

The soil NPK (nitrogen, phosphorus and potassium) sensor uses a physical sensing method [18]. This means that a current or voltage that results from a change in resistance or capacitance, will be measured, like the sensors used for determining temperature or humidity.

All these are embedded into one single device.

4.2.2 RS-485 communication protocol

Such a sensor usually uses the RS-485 protocol to communicate (RS: Recommended Standard). This protocol uses a differential pair, allowing up to 32 drivers per system, unlike the old RS-422 standard which allowed only one driver per system. Moreover, it grants communication up to a distance of 1200 meters and baud rates from 110 Baud to 115200 Baud. There are two possible configurations for this protocol: the 2-Wire one uses a differential pair and is half-duplex, shown in Figure 4.2.2., and the 4-Wire consists of two differential pairs, and it is full-duplex.

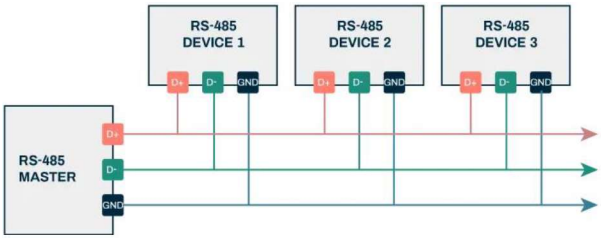

Figure 4.2.2. RS-485 2-Wire configuration [19]

The minimum voltage levels at the driver side are +/-1.5V and the minimum reception levels are +/-200mV, as described in Figure 4.2.3. These voltage levels refer to the difference between the positive and negative lines of the differential pair. In idle conditions, no device is driving the bus, so the level at the receiver side remains undefined. This could lead to interpretation of random data appearing during this period, such as false start bits or interrupts. To overcome this problem, a configuration consisting of pull-up and pull-down resistors at the bus's termination is designed, as outlined in Figure 4.2.4.

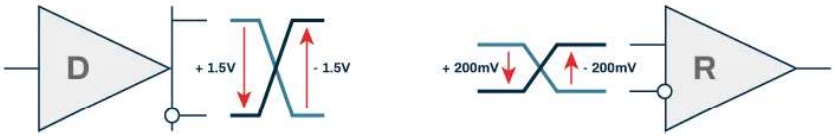

Figure 4.2.3. RS-485 driver minimum voltages and receiver voltage sensitivity [19]

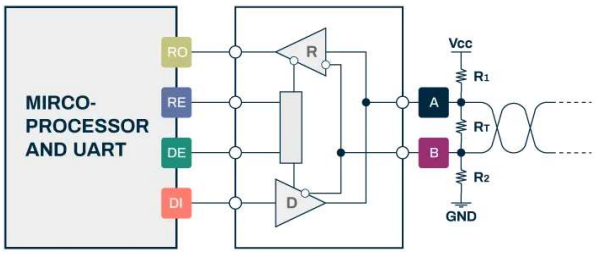

Figure 4.2.4. Fail-safe configuration [19]

The termination resistor has the value of the characteristic impedance of the line, R_T is 120Ω. To determine R_1 and R_2 (R_1=R_2=R) equation (4.2.1) [19] can be used, taking into account that the minimum voltage difference for reception is 200mV. Using this solution only the voltage differences that lie between 0V and 200mV would be seen as logical "1" at the reception and the ones that are between -200mV and 0V would still remain undefined [19]. Communication with the microcontroller is done via the serial port, so a transceiver is needed for the UART (Universal Asynchronous Receiver-Transmitter)-RS485 conversion.

$$V_A - V_B = R_T \frac{V_{CC}}{2R+R_T} \geq 200mV \quad (4.2.1)$$

4.2.3 Optocoupling of the RS-485 line

As the soil quality sensor is connected far away from the board, galvanic isolation is designed in order to prevent any hazards that can appear on the line to get to the microcontroller or the Wi-Fi module. To achieve this, optocouplers are used. As highlighted in Figure 4.2.5., two resistors are needed to be placed at the anode and at the cathode on the LED side of the optocoupler. This improves the CMRR (Common-Mode Rejection Ratio) of the optocoupler, as at these pins parasitic capacitances are present due to the fabrication process. This is the reason why these resistances need to be equal [20].

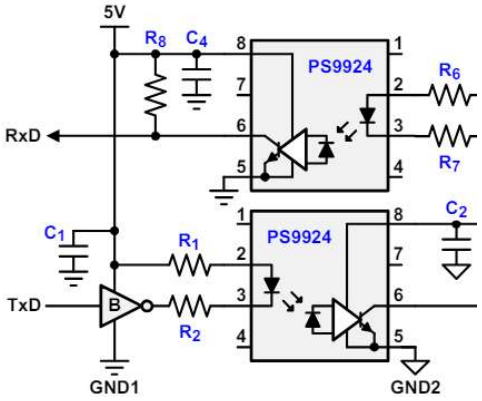

Figure 4.2.5. Optocoupler circuit configuration [20]

4.3 Gas sensor

This sensor is used to monitor the air quality, in particular to measure the concentration of gases such as CO (carbon monoxide), NH_3 (ammonia), NO_2 (nitrogen dioxide) and others. As people also get inside the greenhouse, proper air quality needs to be ensured. For this, concentrations of 9ppm of CO, 0.1ppm of NO_2 and 1ppm of NH_3 don't have to be exceeded [21].

4.3.1 Types of gas sensors

There are several working principles of gas sensors, such as:

A. *Semiconductor based* are made of metal oxide. They work by measuring a change in resistance of the semiconductor material and usually have a heating part for proper functioning. These sensors have low prices, high detection sensitivity, fast response time, but also small linearity range [22].

B. *Electromechanical* sensors measure the current/potential difference/resistance that is generated when the gas oxidizes or is reduced at the electrode due to a chemical reaction. There is also a reference electrode to provide a reference potential and then the output is amplified, as shown in Figure 4.3.1. These sensors have good linearity, resolution up to 0.1ppm, but the sensitivity is strongly affected by the temperature [22].

C. *Infrared* based sensors detect gases based on the absorption of the infrared (IR) radiation by the molecules. An IR lamp emits light in a cell where the gas is present, while at the other end the amount of received light is detected, thus determining the gas concentration. These sensors have fast response, good stability, high accuracy, but narrow range and a high cost [22].

Considering the advantages and disadvantages of all these types of sensors, the most suitable would be the ones based on semiconductors, made of metal oxide. Such a sensor is able to determine the concentration of the three gases mentioned earlier. To interpret the measurement results with the microcontroller, an analog-to-digital conversion is needed, as a resistive value has to be read.

4.4 Control elements

4.4.1 Light control with relay

To be able to control a circuit that powers on and off the light in a greenhouse, a relay is needed. This is a switch that is controlled by a low voltage and controls a high voltage circuit. Figure 4.4.1. shows a scheme of an electromechanical relay. When current flows through the coil, it gets energized, forms a magnetic field and it attracts the contact that will close at the NC (Normally Close) terminal, creating a connection with the COM (Common) terminal. De-energizing the coil means that the contact will close at NO (Normally Open) [23].

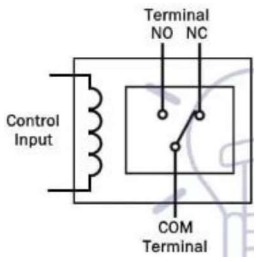

Figure 4.4.1. Electromechanical relay [23]

4.4.2 Water pump control with PWM

A water pump is like a DC (direct current) motor, having typically an inductive and resistive component. To control the water flow a PWM signal is required, adjusting thus the voltage on the pump [25], as depicted in Figure 4.4.2. and calculated with the equation (4.4.1) [24] where δ is the duty cycle.

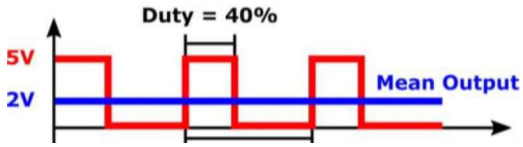

Figure 4.4.2. PWM signal [24]

$$V_{mean} = \delta * V_{max} \tag{4.4.1}$$

The PWM signal generated by the microcontroller cannot be directly applied to the pump, so a circuit configuration with a transistor is needed in order to achieve amplification. Such a design is described in Figure 4.4.3. The PWM signal is applied in the base of the transistor Q1, the resistance having the role to limit the base current. Q1 will amplify this current, turning on the motor. The diode D1 prevents the reverse current generated at switching by discharging the internal inductive element to damage the transistor. To choose the transistor, the maximum collector current needs to be considered, as well as the current gain, the collector-emitter voltage, maximum power and switching frequency [25].

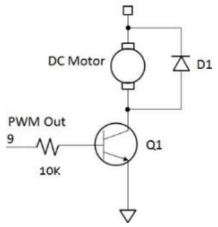

Figure 4.4.3. Circuit configuration [26]

To properly adjust the PWM signal, some external parameters must be taken into consideration, forming thus a control system. Such a system can be built using PID (Proportional – Integrative - Derivative) controller or a Fuzzy Logic one. A Fuzzy Logic controller is not more complex or less effective than a PID one, it also doesn't need tuning that is time consuming. It determines by some computations the needed output based on the current input, not requiring any settling time or dead time [27]. For such applications, a Takagi-Sugeno type 0 Fuzzy system could be a solution.

Figure 4.4.4. depicts the structure of a fuzzy logic system. At the input there are crisp values that are turned into fuzzy values. This means that according to a predefined database that contains fuzzy functions, the membership degree to each of the corresponding set is determined. After that, the activation degree of each rule is computed by using the multiplication (or min) operation between the membership degrees of each set. Defuzzification is then needed, by calculating the weighted average between the activation degree of each rule and its corresponding value [28].

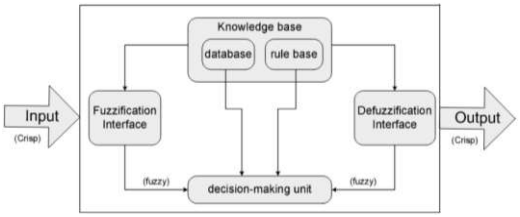

Figure 4.4.4. Structure of a fuzzy system [28]

An example is illustrated in Figure 4.4.5. M_{11}, M_{12}, M_{21}, M_{22} are the fuzzy sets, where M_{i1} and M_{i2} are the fuzzy functions. For certain input values x_1 and x_2 the membership degrees are determined. The rules in this case are: "If x_1 belongs to M_{11} AND x_2 belongs to M_{12} then the output is u_1" and "If x_1 belongs to M_{21} AND x_2 belongs to M_{22} then the output is u_2". "Belongs" refers to the membership degree to the set. For "AND" the minimum operator or multiplication can be used, determining the activation degree of each rule. After that, the weighted average is computed, resulting in the output. A fuzzy partition is a function where the sum of the membership degrees to all the sets is 1.

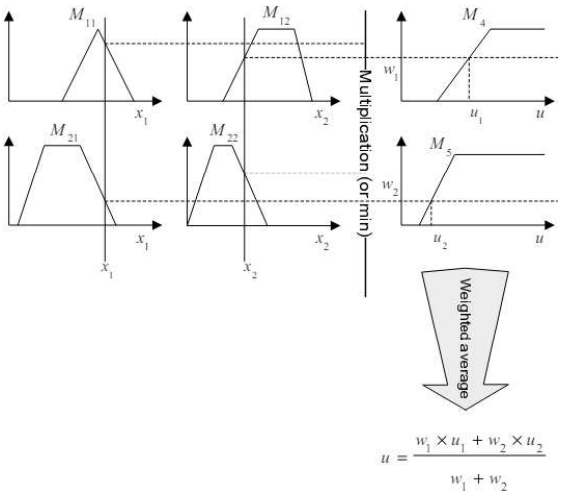

Figure 4.4.5. Determining the output in a fuzzy system [28]

4.5 Water Level sensor

To prevent the functioning of the water pump when there is no water in the tank, a water level sensor is needed. Such a sensor is based on measuring the pressure on the sensor's surface when it is drowned in the water. The pressure is then converted to an electrical signal. Generally, this principle is based on measuring two capacitances, one measured in the air and the other in the liquid, having a common electrode [29].

As this sensor gives at the output an analog signal depending on the level and it will be installed in the galvanically isolated area from the microcontroller, a conversion to digital signal is needed, informing only about the empty and full state of the tank. To achieve this, a hysteresis comparator must be designed. A simple comparator is not a practical solution, as when the voltage level is around the threshold, it can trigger the comparator multiple times.

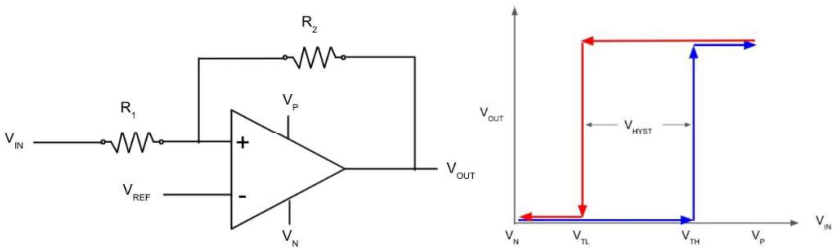

Figure 4.5.1. Hysteresis comparator schematic (left), transfer characteristics (right) [30]

Figure 4.5.1. depicts the schematic of a non-inverting hysteresis comparator and its transfer characteristics, obtained using equation (4.5.1) [30].

$$\begin{cases} V_{TH} = \left(1 + \frac{R_1}{R_2}\right) V_{Ref} - \frac{R_1}{R_2} V_N \\ V_{TL} = \left(1 + \frac{R_1}{R_2}\right) V_{Ref} - \frac{R_1}{R_2} V_P \end{cases} \tag{4.5.1}$$

4.6 The microcontroller

A microcontroller is practically a small computer on a chip composed of a central processing unit (CPU), memory, input/output interfaces and other features like timers, counters, ADCs (analog-to-digital converters). Typically, there are two types of memories in a microcontroller: RAM (random access memory) that is erased at power down and EEPROM (electrically erasable programmable read-only memory) that stores the program to run the microcontroller and that isn't erased when the power supply is disconnected. There is also a non-volatile Flash memory for program storage that is rewritten when programming the device. The main differences between a microcontroller and a microprocessor are presented in Table 4.4.1. [31].

Table 4.4.1. Comparison microcontroller vs microprocessor [31]

Crt. No.	*Microcontroller*	*Microprocessor*
1	Has CPU, memory and auxiliaries	Requires external memories and additional auxiliaries
2	Low-power, suitable in embedded systems, like IoT (Internet of Things)	Consumes more power, suitable in personal computers or data centers
3	Operates at clock frequencies up to 200MHz	Built for demanding computations, can operate at clock frequencies above 1GHz
4	Easy to program, in C, C++ or Python	Configuration requires specialized knowledge
5	Inexpensive	More expansive, more complex

Microcontrollers can be classified based on:

- Core size: 8-bit, 16-bit, 32-bit.
- Architecture: RISC (Reduced Instruction Set Computer) simplifies operations by executing fast simple instructions that are of one-word size, CISC (Complex Instruction Set Computer) executes multiple complex instructions of size higher than one-word in a longer time.
- ARM (Advanced RISC Machines) microcontrollers have modern ARM Cortex subset, increasing reliability and performance, used in automotive and control applications
- PIC microcontrollers are developed by Microchip Technology, and they are the smallest ones in the world, used in robotics, home and industrial applications.
- FPGA-based (field-programmable gate arrays) microcontrollers are used at digital signal processing or high-speed networking, but more important is their versatility as the hardware can be reconfigured depending on the application and that is why they are very expensive [31].

Taking all these into account, a suitable microcontroller for this application would be *ATmega64A*.

4.6.1 ATmega64A

This microcontroller is a low-power AVR 8-bit one. It has advanced RISC architecture, 64kB of Flash program memory and 2kB EEPROM. It can be easily integrated on a PCB, on the grounds of its simple TQFP package. The main features used in this application are the I2C communication (Byte-oriented Two-Wire Serial Interface), the two UART pairs for communication with the Wi-Fi module and the soil quality sensor, the two timers (one for generating PWM signal for controlling the water pump – 8-bit Timer 0 and one for collecting samples for the analog to digital conversion – Timer 2 on 16 bits), as well as the external interrupt for signaling the water tank state and the ADC (10-bit, implemented with SAR – Successive Approximation Register, with the differential channels).

To program the microcontroller, the ICSP (In-Circuit Serial Programming) is used. To perform this a USBASP Programmer is needed. It will facilitate communication with the microcontroller during programming. An advantage of this method is that no extra electronics is needed for programming; the programmer will be directly connected, with a specific header, to the ISP pins of the microcontroller, so, when the microcontroller is soldered on the PCB, access to these pins is crucial. ISP is supported via standard communication protocols (SPI in this case), using the frame of SPI programming, but adding two pins, one for power and one for common ground. To be able to communicate/program through the PC, some driver specific for USBASP needs to be installed [32]. In the case for ATmega64, the pins that act as RX0 and TX0 for UART communication are also used at programming as MISO and MOSI and during programming the RST signal needs to be pulled low. Through the ISP, the internal Flash memory is written, but also the fuse bits and other configurations for the microcontroller to function are accessible.

To connect an external oscillator, capacitors need to be placed on the corresponding clock lines. To dimension these, equation (4.6.1) [33] is used, where C_s is the stray capacitance, a sum of all parasitic capacitances of the IC pins and between the PCB layers, which is usually small compared to the load capacitance C_L, so it can be neglected.

$$C_L = C_S + \frac{C_1 * C_2}{C_1 + C_2} \tag{4.6.1}$$

4.7 The ESP-12E Wi-Fi module

This module has a 32-bit ESP8266 microprocessor with a clock speed of 26MHz. It has a 2.4 GHz Wi-Fi on-board antenna, supporting the IEEE 802.11 b/g/n standards (IEEE 802.11 refers to

the Wi-Fi protocol in the Data Link Layer and the options b/g/n specify the modulation used and other aspects for the encapsulation of data) and uses the TCP/IP (Transmission Control Protocol / Internet Protocol – four-layer protocol) stack. The module also facilitates the UART communication, needed to exchange data with ATmega64 but also for programming. It doesn't need any external components, except the header for programming and buttons to enable the programming mode. From the pinout described in Figure 4.7.1., BOOT (GPIO0) needs to be pulled down while also pulling RST down and after this sequence the module enters automatically to programming mode, the code being fetched from the PC via the UART port. It has a low power consumption, and it is used in low-cost solutions, such as IoT applications, being also easy to implement on the PCB because of its small size [34].

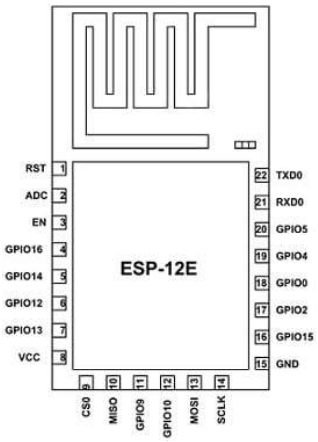

Figure 4.7.1. ESP-12E pinout [34]

The module transmits data via Wi-Fi to a free IoT platform, ThingsBoard, using the MQTT (Message Queuing Telemetry Transport) protocol, specifically for IoT applications. It is a Client-Server publish / subscribe messaging protocol, consisting of MQTT clients that can act as publisher or subscriber of messages and a MQTT broker that handles the communication, as illustrated in Figure 4.7.2. For this application, the clients are the ESP-12E module and the servers of the platform (they interchange the roles, depending on who sends the messages) and the broker is the platform itself, handling the information / telemetries and widgets in the user interface through getter and setter methods. The messages are composed of a topic and a message in JSON (JavaScript Object Notation) format [35]. Sending data from the platform (server) to the device is done via RPC (Remote Procedure Call) [36]. The platform used is ThingsBoard, an open-source one [37].

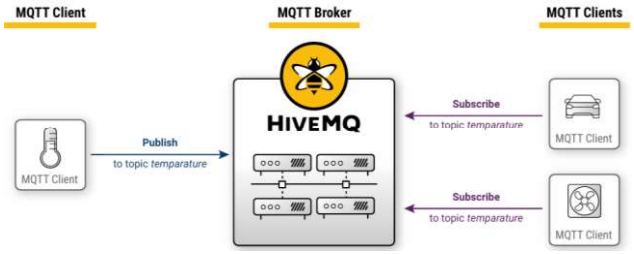

Figure 4.7.2. MQTT Publish/Subscribe architecture [35]

4.8 Power management

As there are two galvanically separated areas on the board, two different power supplies are needed. Consequently, Li-Ion batteries are used to supply each area. These batteries have wide

applications nowadays, being present in phones, laptops and other devices. Inside the battery lithium ions move from anode to cathode when the battery is discharging and in opposite direction at charging through a separator that enables the exchange of the ions. On the outside of the anode and cathode there is a current collector foil facilitating the transmission of the current [38].

In the area where the microcontroller and the Wi-Fi module are located a voltage of 3.3V is needed, while 5V are needed in the galvanically isolated one. Therefore, two different LDOs (Low-Dropout regulators) are needed to provide these voltage levels.

To have a good thermal management for the LDOs (as they have to supply a large amount of current, their junction temperature could get high), thermal vias must be placed below the ICs (integrated circuits) to provide a second heat path through the bottom [39], as shown in Figure 4.8.1. Equation (4.8.1) [40] illustrates the functioning of the thermal circuit and there can be observed that by reducing the thermal resistance of the via cluster (R_{ThVIA}) (the equivalent resistance of the via cluster is the corresponding resistance of each via placed in parallel), the junction temperature (T_J) can be reduced. In the equation, Pd is the dissipated power on the IC, T_A is the ambient temperature (typical value of 25°C), $R_{Th\_JA\_top}$ is the junction to ambient thermal resistance defining the heat path on the top of the IC, taken from the datasheet and R_{ThJC} is the junction-case thermal resistance. At computing the board-ambient thermal resistance (R_{ThBA}), h is the thermal conductivity. To determine the thermal resistance of a via, equation (4.8.2) [40] is used, where λ_{Cu} is the thermal conductivity of the copper, "Length" is the via length and "Area" refers to the plating surface of the via.

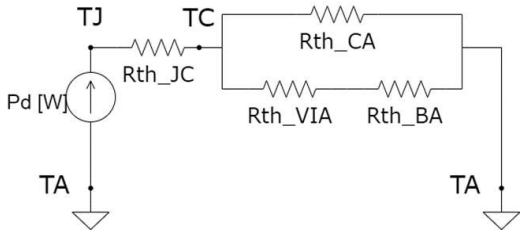

Figure 4.8.1. Thermal circuit

$$P_d = \frac{T_J - T_A}{R_{Th\_JA\_top} \,||\, (R_{Th_{JC}} + R_{Th_{VIA}} + R_{Th_{BA}})};\ R_{Th_{BA}} = \frac{\frac{1}{h}}{Surface}, \frac{1}{h} = 1000\,\frac{°C*cm^2}{W} \quad (4.8.1)$$

$$R_{Th_{VIA}} = \frac{\frac{1}{\lambda_{Cu}} * Length}{Area}\ , \frac{1}{\lambda_{Cu}} = 0.25\,\frac{°C*cm}{W} \quad (4.8.2)$$

5 Implementation

5.1 Block diagram

Figure 5.1.1. presents the block scheme of the whole system and the way the elements interact with the microcontroller. It consists of the ATmega64A microcontroller, the five sensors (gas sensor MICS-6814, air temperature and humidity sensor HS3002, light intensity sensor BH1721, soil quality sensor and water level sensor), the Wi-Fi module ESP-12E, the LCD (Liquid Cristal Display) and the elements for water pump control and light control. The relations between these blocks can be observed on the diagram. Moreover, the galvanic isolation of the components that require 3.3V supply and the ones that need 5V is suggested. All block elements are discussed in what follows.

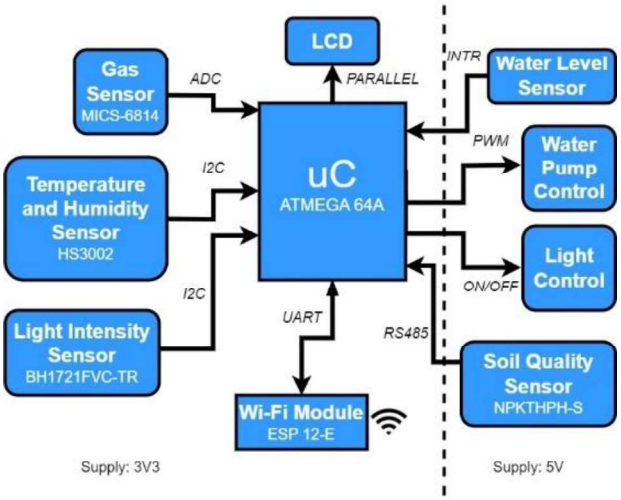

Figure 5.1.1. Block diagram

5.2 ATmega64A

The microcontroller, depicted in Figure 5.2.1. with all the external components, needs 3.3V for proper functioning with a low power consumption and thus, to keep the device in its SOA (Safe Operating Area), the clock frequency needs to be set at 8MHz [41]. An external oscillator is connected, ECS-80-10-33-CHN-TR3, having a load capacitance of 10pF, a tolerance of +/-10ppm and stability of +/-25ppm [42]. External capacitors need to be connected at both pins of the oscillator to sustain a proper clock signal. Their values are determined by using equation (4.6.1), resulting in a value of 20pF for C_3 and C_4.

For the I2C bus, the values of the pull-up resistors R_2 and R_3 are calculated using equation (4.1.2), as the chosen serial clock frequency is 100kHz, and thus a value of 2.2kΩ respects the inequality and ensures low consumption. A logic "AND" is performed by the rectifying diodes D_3 and D_4 together with the pull-up resistor R_5 to control the data flow into the PE0 pin that is used as well as RX0 pin in the UART communication with the soil quality sensor, as well as input program pin during programming. For the PE1 pin multiplexing isn't needed as in idle state the RS485 transceiver will be in transmission mode, so the received data is ignored. For the proper functioning of the ADC the supply AVCC has to be connected via a second order low-pass filter (with the cut-off frequency of 159kHz) constructed with LC elements in order to achieve a stable voltage. This

voltage is also used as reference voltage for the conversion whereas the AREF pin is decoupled to ground to improve the noise immunity of the converter [41]. Also, the data lines for the LCD are on port C and the control lines are on pins PA5÷7.

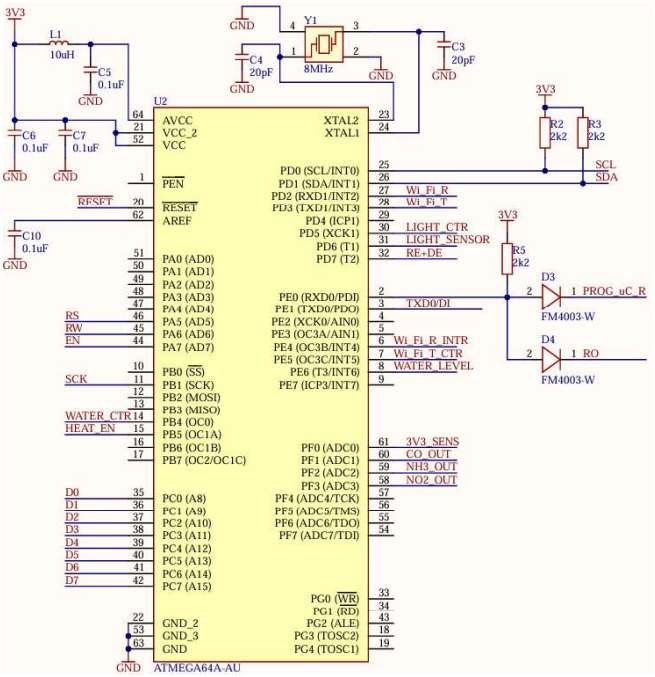

Figure 5.2.1. ATmega64A with the external components

5.3 ESP-12E module

Figure 5.3.1. illustrates the necessary connections for proper functioning of the Wi-Fi module. For normal functioning GPIO15 needs to be pulled low and EN high, having the role of chip enable. This is performed by resistors of values 10kΩ, value chosen in order to limit the current consumption to minimum. To enter programming mode GPIO0 that acts as BOOT signal and the RESET have to be pulled low, these being controlled via pushbuttons [43]. A decoupling capacitor of 10μF is necessary to be placed close to the module in order to supply it when a Wi-Fi transmission or reception is performed as these operations require much current (for instance, a transmission using the 802.11b standard with a maximum data rate of 11Mbps requires a current consumption of 170mA). The capacitor acts thus like a local energy reservoir. The communication with ATmega64 is performed via the GPIO5 and GPIO4 by using the SoftwareSerial feature with a baud rate of 9600.

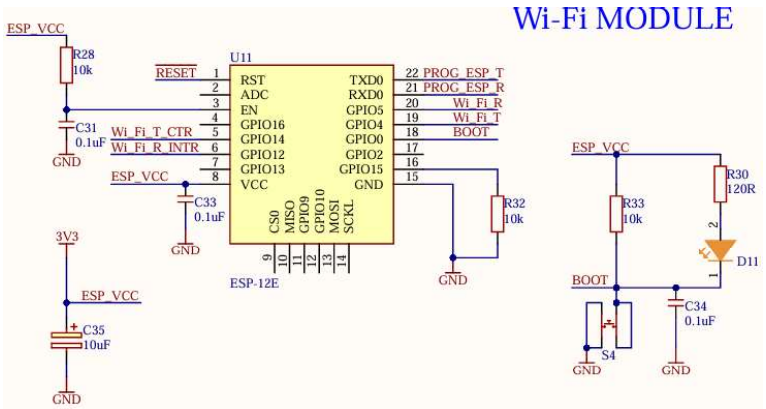

Figure 5.3.1. ESP-12E module with its external components

5.4 Design for RS-485 communication

Figure 5.4.1. shows the configuration for the RS-485 communication, with the transceiver and the optocouplers for the galvanic isolation. By applying equation (4.2.1) for the fail-safe configuration, it results that the resistors R_{11} and R_{19} need to be of 1.43kΩ, while R_{15}, the termination resistor, 120Ω. For example, if it appears a voltage of 50mV on AB_P line, so on AB_N it would be -50mV, the differential voltage would be 100mV, but this configuration would force a current flow (3.46mA) from R_{11} to R_{19} so that a voltage drop of more than 200mV (minimum voltage necessary to be detected; in this example, 415mV) would appear across R_{15}. This is not the case though when there is a voltage of -50mV on AB_P (so 50mV on AB_N), as there could be a current generated on the R_{11} side in the direction of R_{15} but there would be not return path for it because it could not flow from negative potential to a positive one. These explanations are made in the ideal case where no current flows on the bus (assuming there is high impedance at both ends). The diodes D_6, D_7, D_8, D_9 ensure that no voltage above 5V or below 0V gets to the MAX483 transceiver that works as follows: when /RE and DI are "0", the RO output is enabled and data is transmitted to the microcontroller, to the RXD0 pin; when they are put in "1", the DI input is enabled, and data is put on the bus [44].

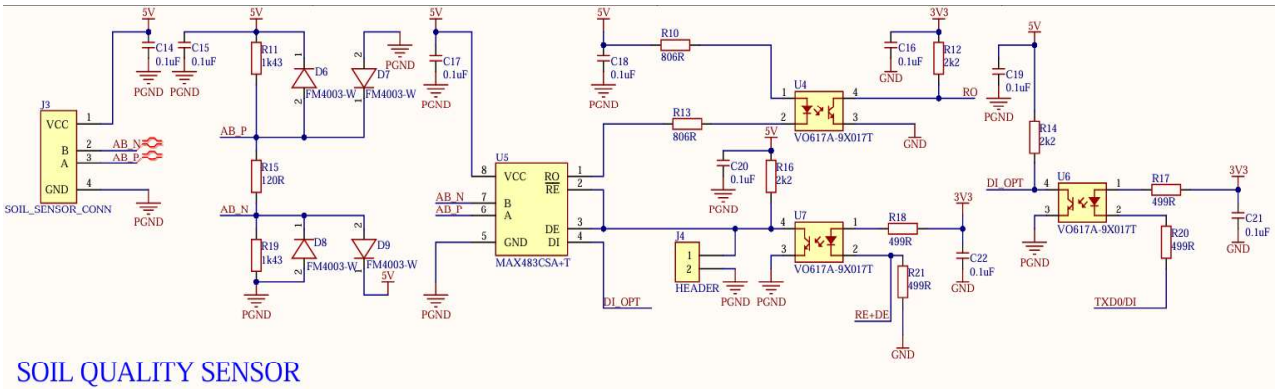

Figure 5.4.1. RS-485 circuit design and optocoupling

For the optocoupling part, at RO, there needs to be considered that in the worst case the voltage for transmitting "0" is 0.4V and "1" 4.6V (for a supply voltage of the transceiver of 5V) and that the functioning of the optocouplers shouldn't consume much power, the design relations for the resistances (whose role is explained in section 4.2.3.) from equation (5.4.1) were used, where I_F is the forward current through the LED, imposed at 2.1mA, V_F is the forward voltage drop on the LED taken from the characteristic of the optocoupler, V_{CE} is the corresponding collector-emitter voltage on the transistor side and I_C is the collector current of the transistor [45]. A value of 806Ω was chosen for these resistances after consulting the distributor availability. To determine the value of the resistance on the transistor side, whose role is to diminish the power dissipated on the transistor, equation (5.4.2) was utilized and a value of 2.2kΩ was chosen. Similarly, the optocoupling circuit for /RE, DE, and DI are designed. The optocoupler has an isolation voltage of 5.3kV.

$$I_F = 2.1mA \Rightarrow \begin{cases} V_F = 1.1V \\ V_{CE} = 0.2V \\ I_C = 1.5mA \end{cases} \Rightarrow R_{10} = R_{13} = \frac{5V-1.1V-0.4V}{2*2.1mA} = 833\Omega \quad (5.4.1)$$

$$R_{12} = \frac{3.3V-0.2V}{1.5mA} = 2.07k\Omega \quad (5.4.2)$$

Figure 5.4.2. shows two simulations done in LTspice for the optocoupled circuit for the RO voltage and the corresponding current. There can be observed that the RO voltage at the output of the

transceiver is in phase with the one at the output of the optocoupler, having a value for logic "1" of 3.2V that can be correctly interpreted by the microcontroller. Also, the current through the LED that turns on when a logic "0" needs to be transmitted, is about 2.2mA while the collector current of the transistor is about 1.5mA. Similar values are obtained for the other two circuits.

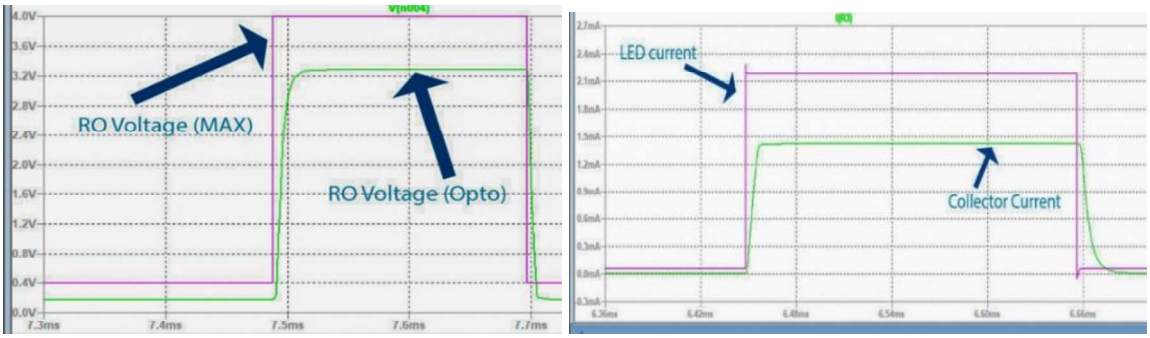

Figure 5.4.2. Simulations for voltage (left) and current (right)

To communicate with the soil quality sensor, first a string of bytes needs to be sent from the microcontroller via UART0 (with a baud rate of 4800) which is passed through the transceiver intro RS-485. The string from Figure 5.4.3. signalizes the sensor that it needs to perform a measurement. /RE/DI pin needs to be put in "1" to send these bytes and after that in "0" to be able to read the sensor's response that consists of a string of bytes as depicted in Figure 5.4.4., where the humidity is expressed in "%", temperature in "°C", nitrogen, phosphorus and potassium in "mg/kg" [46].

Address	Function Code	Start Address (Hi)	Start Address (Lo)	Number of Points (Hi)	Number of Points (Lo)	Error Check (Lo)	Error Check (Hi)
0x01	0x03	0x00	0x00	0x00	0x07	0x04	0x08

Figure 5.4.3. String of bytes sent by the master [46]

Address	Function Code	Number of byte	humidity	temperature	conductivity	PH	N	P	K	Error Check (Lo)	Error Check (Hi)
0x01	0x03	0x0E	0x01 0xD0	0x01 0x4C	0x00 0x2C	0x00 0x5A	0x00 0x20	0x00 0x58	0x00 0x68	0x70	0x29

Figure 5.4.4. String of bytes sent by the sensor as response [46]

5.5 Gas sensor

Figure 5.5.1. shows the MICS-6814 gas sensor with the necessary resistances for proper heating and sensing. For dimensioning the resistances from the heating part, relations (5.5.1) [47] are used considering the necessary heating voltages and currents for each type of gas taken from datasheet and that $R_{DS,on}$ of Q_1 p-type MOS transistor is negligible, about 0.6Ω, supporting a maximum drain current of 330mA [48]. This type of transistor was chosen because it has a lower resistance in on-state than a BJT. The values for the resistances were chosen by taking into account the price and availability. To enable heating the signal HEAT_EN needs to be pulled low by the microcontroller for a minimum period of 30s. After that, for a period of about 150-180s the heating is disabled as this process requires much current (about 88mA). The value of 56kΩ for the sensing resistances has been taken from the datasheet required in order not to destroy the sensitive layer [47]. To determine the sensing resistance, first the current needs to be determined by measuring the voltage on the 56kΩ resistance, then the voltage on R_s and by dividing this voltage by the current, the value of R_s is determined.

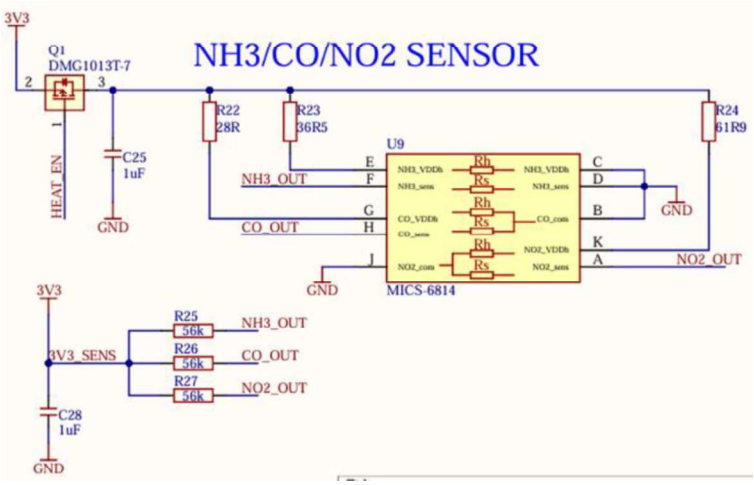

Figure 5.5.1. MICS-6814 sensor with its external components

$$\begin{cases} V_{H\_CO} = 2.4V;\ I_{H\_CO} = 32mA \\ V_{H\_NO2} = 1.7V;\ I_{H\_NO2} = 26mA \\ V_{H\_NH3} = 2.2V;\ I_{H\_NH3} = 30mA \end{cases} \rightarrow \begin{cases} R_{22} = \frac{3.3V-2.4V}{32mA} = 30\Omega \\ R_{24} = \frac{3.3V-1.7V}{26mA} = 61.5\Omega \\ R_{23} = \frac{3.3V-2.2V}{30mA} = 36.6\Omega \end{cases} \quad (5.5.1)$$

Knowing that the sensing resistance for CO is between [100kΩ; 1500kΩ], for NH_3 [10kΩ; 1500kΩ] and for NO_2 [0.8kΩ; 20kΩ] [48]. Consequently, a resolution of 1kΩ for CO and NH_3 measurements and one of 0.1kΩ for the NO_2 would be enough to get a reliable result. The resolution of the ADC is 3.22mV ($3.3V/2^{10}$) and by taking the extreme cases, 0.8kΩ sensing resistance with a current of 0.058mA (3.3V/56.8kΩ) would result in a sensing voltage of 46.4mV and a voltage drop across the 56kΩ resistance of 3.248V (At 0.9kΩ would result a sensing voltage of 52.19mV, at a difference in voltage large enough from the previous case). This voltage drop is determined by using the differential channels property of the ADC. Taking the case R_s equals 1500kΩ would result in a current of 0.0021mA, a sensing voltage of 3.18V and a differential voltage of 0.12V. It can be observed that all voltages are greater than the resolution of the ADC, so they can be detected. In the code (written in C using Microchip Studio environment) to determine the resistance voltage samples are taken for each gas (20 differential voltage samples to determine the current and 20 samples of the sensing voltage). This process starts only after the heating time is over and the transistor is turned off. Timer 2 is used in CTC (Clear Timer on Compare Match) mode. By writing into the OCR2 register the value 255 and having a prescaler factor of 1024, a frequency of 15.26Hz is determined by using the equation from the datasheet. Also, an interrupt on Compare Match is enabled, thus at every 66ms a sample from the ADC is read, till all the samples for the three types of gases are known. The ADC needs to be configured at every occurrence of the interrupt, as the channels that give the sample value are multiplexed. In consequence, the converter is started after configuring the needed channel, then the sample is read, and the converter is stopped. After reading all the samples, their values are averaged and the voltage computed by using equation (5.5.2), where N is the sample's value.

$$V = \frac{V_{Ref*N}}{2^{10}} \quad (5.5.2)$$

Figure 5.5.2. illustrates some simulations done in Proteus 8.13 for the case where NO_2 concentration is determined using channel 3 of the ADC; in the left the result of the conversion for 0.8kΩ sensing resistance (which corresponds to 0.77kΩ, first byte being the integer part and the second byte the decimal one, in hexadecimal format) and right for 20kΩ resistance, the conversion result being 20.1kΩ, a slight variation that would not impact the gas concentration significantly.

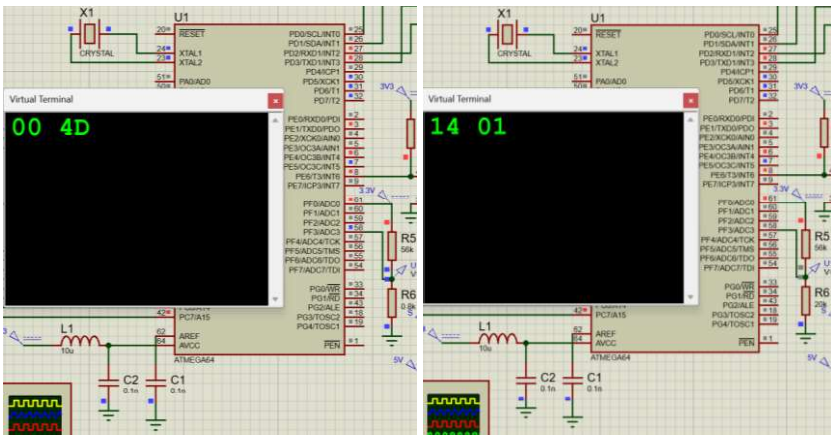

Figure 5.5.2. Simulation results for 0.8kΩ (left) and 20kΩ (right)

Figure 5.5.3. show conversion results for determining CO and NH_3 concentrations (channel 1 and 2 of the ADC) for 100kΩ sensing resistance (left), where the two bytes represent the integer value of the resistance. In the right there is the coversion result of 1514kΩ for a resistance of 1500kΩ.

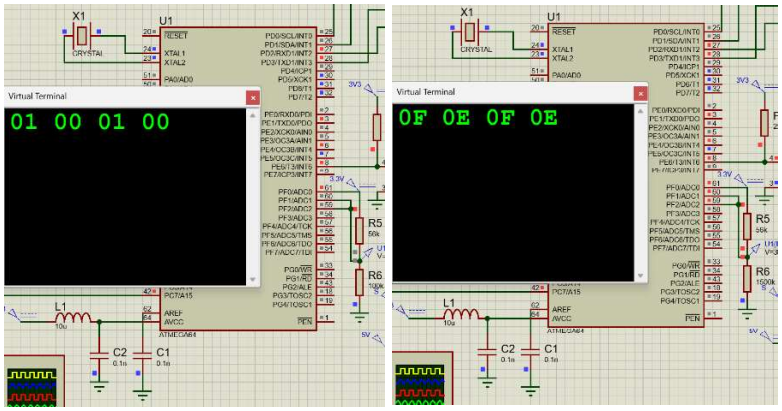

Figure 5.5.3. Simulation results for 100kΩ (left) and 1500kΩ (right)

After determining the values for the sensing resistances, the corresponding gas concentration has to be computed. By analyzing the three characteristics from Figure 5.5.4., there can be observed that the dependency between the resistance and gas concentration isn't linear as the abscissa is represented by values in logarithmic scale, so a linearization on intervals needs to be made.

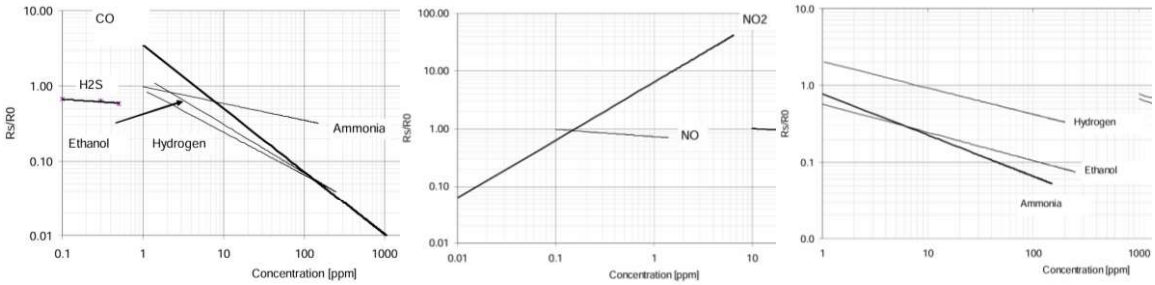

Figure 5.5.4. R_s/R_o vs ppm characteristic for CO (left), NO_2 (middle), NH_3 (right) [47]

Figure 5.5.5. shows the linearized characteristics for the three types of gas concentrations, done using Excel. To implement these in the microcontroller, the relations from (5.5.3) and (5.5.4) are applied, representing the line equation for each interval, where R_o is the sensing resistance in clean air which has to be experimentally determined.

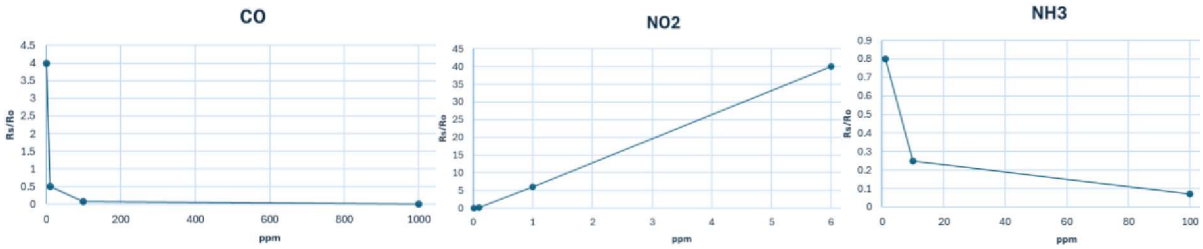

Figure 5.5.5. R_s/R_o vs ppm linearized characteristic for CO (left), NO_2 (middle), NH_3 (right)

$$CO:\begin{cases} -15000\frac{R_S}{R_O}+1150; \frac{R_S}{R_O}\le 0.07 \\ -209\frac{R_S}{R_O}+114; \frac{R_S}{R_O}\le 0.5 \\ -2.5\frac{R_S}{R_O}+11; \frac{R_S}{R_O}\le 4 \end{cases} ; NO_2:\begin{cases} 0.9\frac{R_S}{R_O}-0.035; \frac{R_S}{R_O}\le 0.15 \\ 0.153\frac{R_S}{R_O}+0.07; \frac{R_S}{R_O}\le 6 \\ 0.15\frac{R_S}{R_O}+0.09; \frac{R_S}{R_O}\le 40 \end{cases} \quad (5.5.3)$$

$$NH_3:\begin{cases} -500\frac{R_S}{R_O}+135; \frac{R_S}{R_O}\le 0.25 \\ -16\frac{R_S}{R_O}+14; \frac{R_S}{R_O}\le 0.8 \end{cases} \quad (5.5.4)$$

5.6 Temperature / Humidity sensor and Light Intensity sensor

Figure 5.6.1. illustrates the connections for the temperature and humidity sensor HS3002, that has a resolution of 0.01%RH, respectively 0.015°C, and the light intensity sensor BH1721, having a resolution of 1÷8 lx, to the I2C bus. The addresses of the sensors are 44h (1000100b) for HS3002 and 23h (0100011b) for BH1721. Following the standard presented in chapter 4.1.4., first the start bit is sent, then the address to whom the write bit ("0") is appended and the sensor will send an acknowledge bit ("0"), followed by the stop bit. To require data from the sensor, the start bit is put on the bus, after that to the address the read bit is appended ("1") and acknowledgement from the sensor is issued, followed by the required data. For the HS3002 sensor, 32 bits of data are sent. The first two most significant ones signalize if the data is valid (if they are "00", it means valid data), the next 14 bits are the data representing the relative humidity, followed by 14 bits for temperature and the two least significant bits which need to be masked [49]. At BH1721, to start the communication, first the DVI pin needs to be pulled high after a minimum period of 1µs after powering the sensor on. Also, after sending the address plus the write bit, the byte for the resolution mode has to be sent. For auto-resolution mode, 00010000, meaning that for a light intensity below 4000lx, the high-resolution mode is selected (1 lx), else the low-resolution is preferred (8 lx). The data representing the value of the light intensity is on 16 bits and to get the result, this value has to be divided by 1.2 [50].

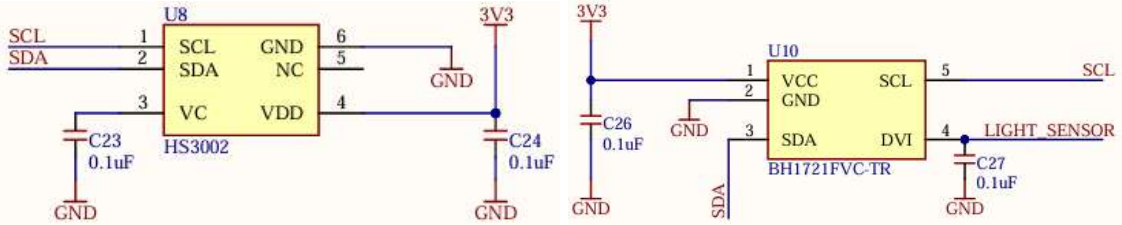

Figure 5.6.1. HS3002 sensor (left) and BH1721 sensor (right)

Figure 5.6.2. illustrates a simulation in Proteus with the help of the I2C debugger for the HS3002 sensor, consisting of the addressing part (start bit, address and write bit, acknowledge and stop bit) and measurement part (start bit, address and read bit, the four data bytes and stop bit).

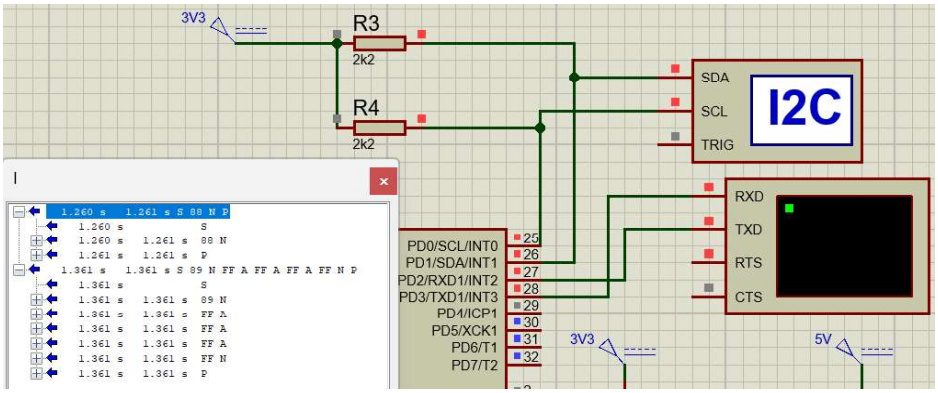

Figure 5.6.2. I2C debugger simulation

$$Humidity[\%] = \frac{Humidity[13:0]}{2^{14}-1} * 100 \quad (5.6.1)$$

$$Temperature[°C] = \frac{Temperature[15:2]}{2^{14}-1} * 165 - 40 \quad (5.6.2)$$

Equations (5.6.1) and (5.6.2) are used to determine the values of the temperature and humidity based on the data bits received from the sensor [49].

5.7 Water Pump control and Light control

The circuits to control the water pump and light are also designed with optocouplers as the control elements are connected far away from the board. Figure 5.7.1. shows the control circuit for water pump. Equation from relation (5.4.1) is used to design the resistances from the LED side, taking into account that a larger current is passed through this side (4mA) as a continuous current of about 300mA is needed for proper functioning of the pump [51]. On the transistor side there is a collector current of about 5mA (and a collector-emitter voltage of 0.4V) that is amplified by another n-type bipolar transistor (that has a current transfer ratio h_{fe} of 100) to reach the desired current. As the pump is controlled using PWM signal of 120Hz (experimentally set, in order to reduce the switching losses in the circuit and keep a uninterrupted functioning of the pump), the circuit elements need to be able to switch with the frequency of this signal. Q_2 having a transition frequency f_T of 100MHz that corresponds to a bandwidth of about 1MHz is suitable, as well as the n-MOS transistor Q_3 that has a 0.12Ω resistance in on-state and rise and fall times appropriate for the PWM frequency [52]. At its gate the PWM signal generated by using Timer0 is fed. The pump resembles an inductive element in series with a resistive one. So, at switching, the inductive element must be discharged. To facilitate this, a diode is placed in parallel with the pump. A Schottky diode has been chosen due to its low reverse recovery time, the ability to evacuate charges in a small time at switching from on state to off state. To dimension the value of the resistance R_{31} equation (5.7.1) is used.

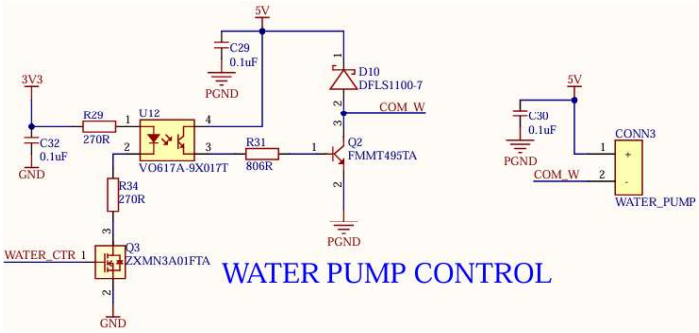

Figure 5.7.1. Water pump control circuit

$$R = \frac{5V - V_{CE} - V_{BE,on}}{I_C} = \frac{5V - 0.4V - 0.6V}{5mA} = 800\Omega \quad (5.7.1)$$

Figure 5.7.2. depicts simulations made in LTspice on how the collector-emitter voltage of the Q_2 transistor behaves when there is no diode in parallel with the load and there is an overshoot that gets to around 100V, while when there is a diode, the overshoot disappears.

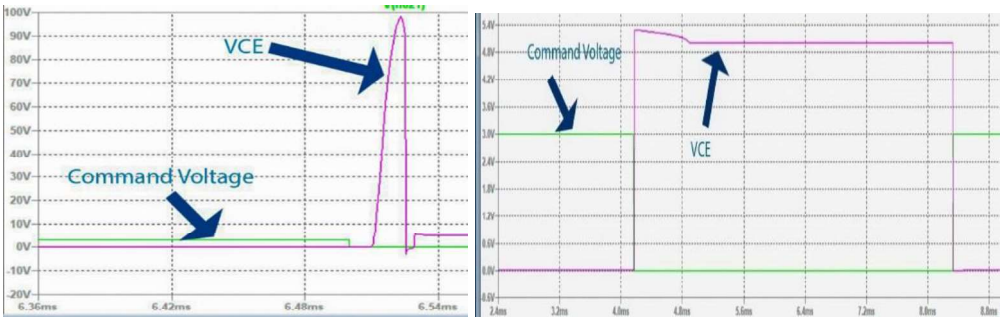

Figure 5.7.2. Simulation results without Schottky diode (left) and with Schottky diode (right)

Figure 5.7.3. shows a power peak of 480mW that appears for a small period of time, at switching, and that can be handled by the transistor as its absolute maximum power dissipation is 500mW.

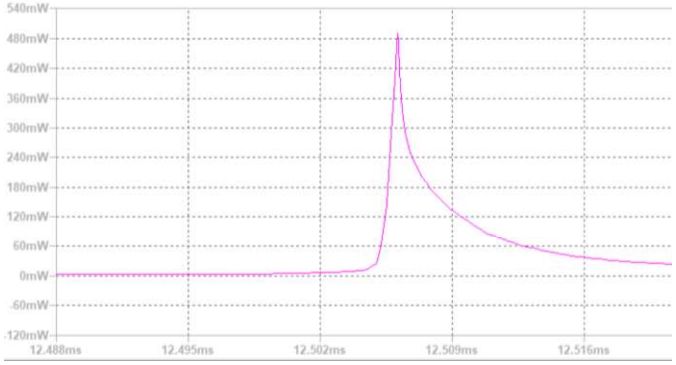

Figure 5.7.3. Power dissipated on Q_2

To control the duty cycle of the PWM signal, a fuzzy inference system is developed which computes the value of the duty cycle based on the air temperature and soil humidity. The system is built using the FuzzyLogicDesigner application from Matlab. As Figure 5.7.4. shows, a Takagi-Sugeno system of degree 0 is developed, having 3 fuzzy sets for temperature and humidity and at the output 4 sets that are singleton representing the value of the duty cycle.

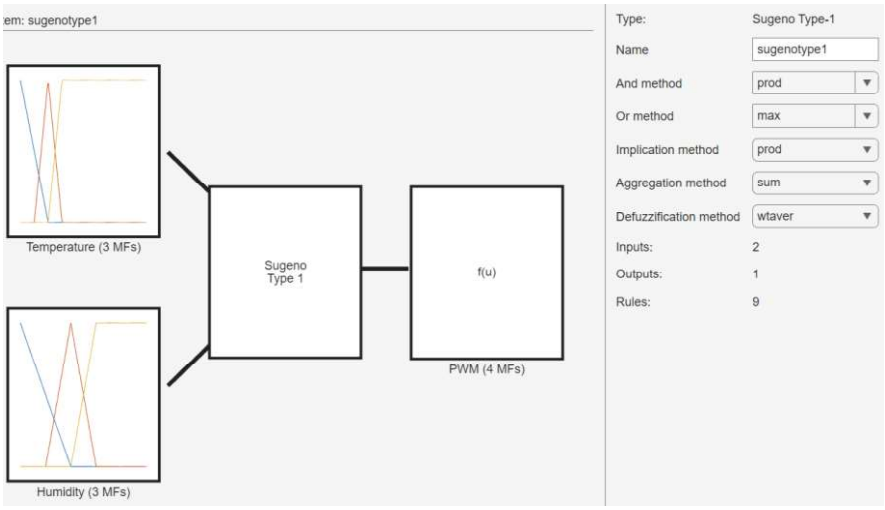

Figure 5.7.4. Structure of the Takagi-Sugeno system degree 0

Figure 5.7.5. illustrates how the fuzzy sets look like for the two input variables. As such, for the temperature there are two triangular sets, “Small” and “Medium”, and a trapezoidal one, “High”, and for the humidity the same, “Dry” and “Fine” are triangular and “Wet” is trapezoidal. The desired temperature and humidty (the value where the “Medium” and “Fine” triangle have a degree of membership equal to 1) can be adjusted, keeping the same shape for the middle sets and narrowing or enlarging the other ones. The PWM output function has the following singletone values: 0, 25, 50, 100.

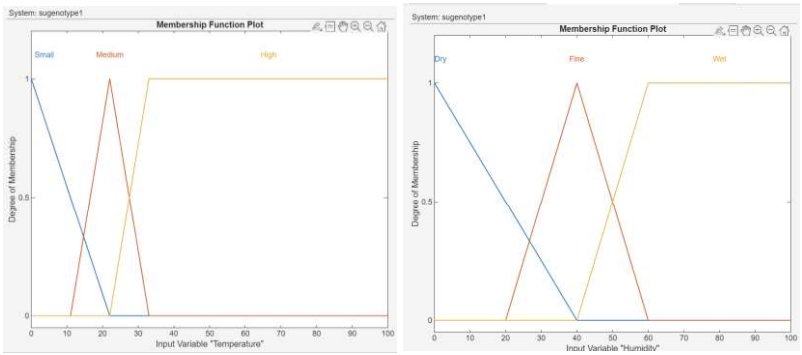

Figure 5.7.5. Fuzzy sets for temperature (left) and humidity (right)

To further define the system, the rule base has to be specifyed. Figure 5.7.6. describes the rule base for this system (left) and the control surface (right) based on the rules. For small temperature and humidity, the duty cycle is 100% and for high temperature and humidty the duty cycle is 0. For the “AND” operator the multiplication is used and the process described in chapter 4.4.2. is carried on.

	Rule	Weight	Name
1	If Temperature is Small and Humidity is Dry then PWM is 100	1	rule1
2	If Temperature is Medium and Humidity is Dry then PWM is 50	1	rule2
3	If Temperature is High and Humidity is Dry then PWM is 25	1	rule3
4	If Temperature is Small and Humidity is Fine then PWM is 50	1	rule4
5	If Temperature is Medium and Humidity is Fine then PWM is 25	1	rule5
6	If Temperature is High and Humidity is Fine then PWM is 0	1	rule6
7	If Temperature is Small and Humidity is Wet then PWM is 0	1	rule7
8	If Temperature is Medium and Humidity is Wet then PWM is 0	1	rule8
9	If Temperature is High and Humidity is Wet then PWM is 0	1	rule9

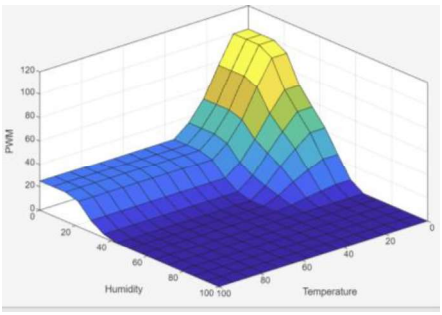

Figure 5.7.6. Rule base (left) and control surface (right)

For the light control a relay is used, as depicted in Figure 5.7.7., in order to make only a simple switch between the on state and the off state for a circuit that will be connected to the board. To control the state of the relay, the circuit is designed similar to the previous one, only that a current of 2.1mA is passed through the LED and 1.5mA for the collector current on the transistor side. Dimensioning R_{39} is done by using equation (5.7.1) with the collector-emitter voltage of 0.2V. As the relay has an inductive element that does the switching, a diode is inserted in parallel with it.

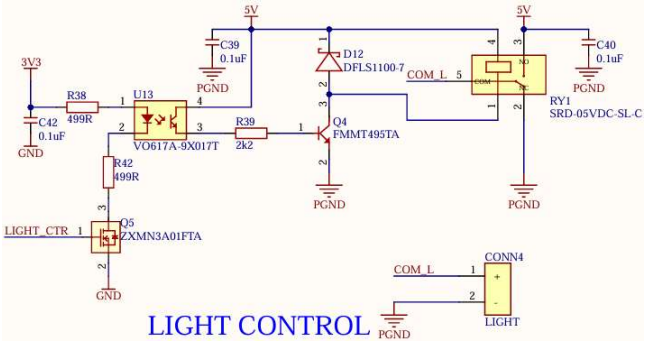

Figure 5.7.7. Light control circuit

5.8 Water Level sensor

The water level sensor, giving at the output analog voltages [53], a digitization in form of "full" and "empty" state of the tank is performed. This means that an interrupt signalizing the state of the water tank will be triggered by the output of the hysteresis comparator that is fed through the optocoupler to the microcontroller, as illustrated in Figure 5.8.1. The thresholds for the comparator are experimentally set at 2.1V and 2.3V. By using equation (4.5.1), the relations from (5.8.1) for dimension the resistances for the hysteresis comparator are obtained, getting the reference voltage from a voltage divider between R_{44} and R_{43}. The values of the resistances for the optocoupling part, R_{36}, R_{37}, R_{40} are determined by using relations (5.4.1) and (5.4.2).

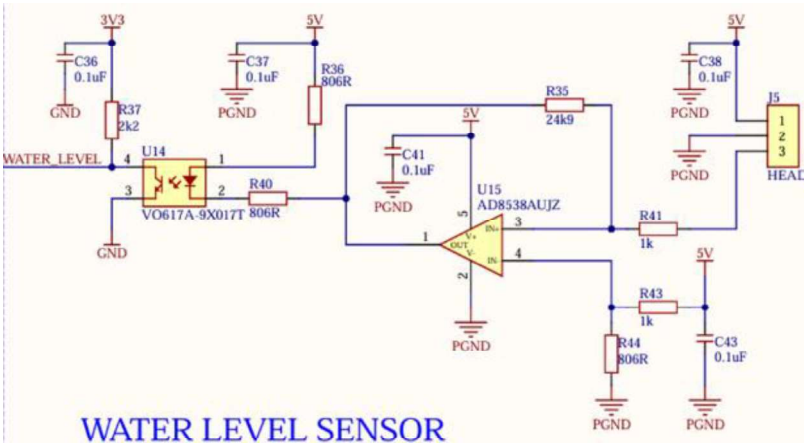

Figure 5.8.1. Circuit for the water level sensor

$$\begin{cases} 25R_{41} = R_{35} \\ 0.79R_{43} = R_{44} \end{cases} \tag{5.8.1}$$

The chosen operational amplifier AD8538 has a very low offset voltage of maximum 13µV, and a low input bias current of 25pA, thus not influencing the output voltage in a significant way. Also, it has a high CMRR of 150dB, reducing the influence on the output of the common mode voltage at the input and a PSRR (Power Supply Rejection Ratio) of 125dB, rejecting any noise that can appear on the supply line. AD8538 supports also a single-supply operation, as such it is able to function when supplied by 0V and 5V [54].

Figure 5.8.2. represents a simulation in LTSpice of the circuit where the abscissa acts as the input voltage from the sensor and on the ordinate is the voltage that gets to the microcontroller (WATER_LEVEL). The two thresholds are at 2.05V and 2.37V, which are close to the imposed ones and don't alter the result.

Figure 5.8.2. Transfer characteristic

Figure 5.8.3. shows a simulation in Proteus of the circuit connected to the microcontroller of the external interrupt functioning on the falling edge, signalizing that the tank is empty. There can be observed that the PWM signal (on channel A of the digital oscilloscope) which controls the water pump is stopped and the corresponding pin is pulled low to stop the pump.

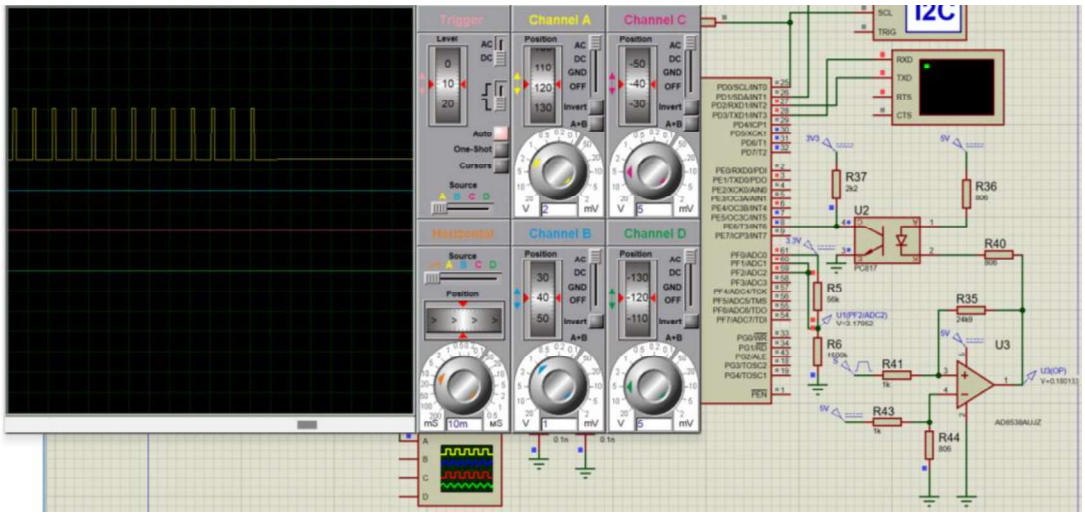

Figure 5.8.3. Functioning of the external interrupt

5.9 Power supplies

To ensure that a voltage of 3.3V is applied to the side of the board where the microcontroller, the Wi-Fi module, the I2C sensors, the gas concentration sensor and the LCD are located, an LDO is used, as illustrated in Figure 5.9.1. (upper side). The 5V LDO, represented in Figure 5.9.2. (lower side) supplies the circuits for the soil quality sensor, water level sensor, water pump control and light control. Each of the ICs are powered up by two Li-Ion batteries, resulting in a voltage of 7.4V and a current capability of 6.8Ah. The bypass capacitors at the input of the LDOs have the role to filter out the noisy component of the signal and the ones at the output help the transient response, reducing the rise/fall times and overshoot. The values for these are taken from the datasheet of the two ICs. The LEDs at the output signalize if the board is supplied. A current of approximately 10mA flows through them and there is a voltage drop of about 2V, resulting in the necessary values of the resistances in the anode depicted in the figure below.

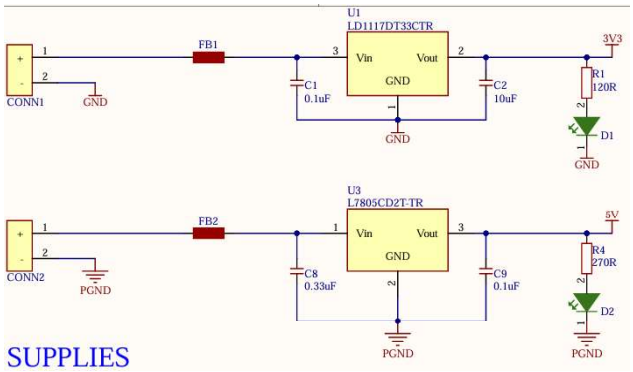

Figure 5.9.1. Circuit configuration for the two LDOs

The maximum current that has to be delivered by the 3.3V LDO is expressed in Table 5.9.1., resulting in a total current of about 850mA. The power dissipated on the LDO is determined by the equation (5.9.1), where V_{outLDO} is 3.3V and I_{out_max} is 850mA, resulting in a power of 3.48W. From

the datasheet, R_{ThJC} is 8°C/W and R_{ThJA} 100°C/W [55]. The thermal resistance of one via, having the hole diameter 0.3mm and pad diameter of 0.7mm, (R_{ThVIA}) placed under the IC is 110°C/W (computed using Saturn Toolkit [56], see Appendix 9.4.). By placing 9 vias on the area under the IC at a distance of 2.1mm from each other and computing the junction temperature using equation (4.8.1), a temperature of 106°C is obtained, less than the maximum junction temperature of 125°C specified in the datasheet.

$$P_d = (7.4V - V_{outLDO})I_{out\_max} \tag{5.9.1}$$

Table 5.9.1. Maximum current consumption for the components supplied by 3.3V

Crt. No.	*Component*	*Max current consumption [mA]*
1	Green LED	10
2	ATmega64A	400
3	LCD1602	140
4	Reset LED + Boot LED	20
5	HS3002	24.4µ
6	BH1721	200µ
7	Heating MICS-6814	88
8	Sensing MICS-6814	0.12
9	ESP-12E	170
10	5 optocouplers	10.5
11	1 optocoupler (for water pump control)	4
TOTAL		842.84

Table 5.9.2. explains the maximum current delivered by the 5V LDO, 450mA. By applying equation (5.9.1), it results a total power dissipation of 1.08W on1 the IC. Knowing that R_{ThJC} is 3°C/W, R_{ThJA} 62.5°C/W [57] and R_{ThVIA} is the same as above, and applying equation (4.8.1), a junction temperature of 52.1°C is obtained, less than the maximum specified value of 125°C.

The junction temperatures for both ICs were also compared with an online tool for computing the thermal parameters for devices placed on PCBs [58].

Table 5.9.2. Maximum current consumption for the components supplied by 5V

Crt. No.	*Component*	*Max current consumption [mA]*
1	Green LED	10
2	Water Pump	250-300
3	MAX483	400µ
4	Soil Quality sensor	100
5	Water level sensor	20
6	5 optocouplers	7.5
7	1 optocoupler (for water pump control)	5
8	Operational Amplifier from water level sensor circuit	180µ
TOTAL		450

5.10 PCB design

The PCB for all the circuit elements presented above is developed using Altium. The stack-up for the board is shown in Figure 5.10.1., one that has been chosen in accordance with the manufacturer's possibilities, JLCPCB [59]. The traces are routed on the top and bottom layers, while the second layer is for the ground plane and the third for the power distribution.

#	Name	Material	Type	Weight	Thickness
	Top Overlay		Overlay		
	Top Solder	Solder Resist	Solder Mask		0.01mm
1	Top Layer	CF-004	Signal	1oz	0.035mm
	Dielectric 2	PP-023	Prepreg		0.2104mm
2	GND	CF-004	Signal	1/2oz	0.0175mm
	Dielectric 1	Core-023	Core		1.065mm
3	POWER	CF-004	Signal	1/2oz	0.0175mm
	Dielectric 3	PP-014	Prepreg		0.2104mm
4	Bottom Layer	CF-004	Signal	1oz	0.035mm
	Bottom Solder	Solder Resist	Solder Mask		0.01mm
	Bottom Overlay		Overlay		

Figure 5.10.1. Stack-up

The layers are presented in Appendix 9.2. The gas concentration sensor, connected to the ADC, is placed close to the microcontroller to reduce the impact of the impedance of the traces for the measurement. Also, the traces for connecting the oscillator are as short as possible so that the rise and fall time of the clock signal isn't significantly affected. Guarding is designed for the lines that carry signal with a higher frequency than the others. These are the I2C (SDA and SCL) and the clock (connection between the oscillator and the microcontroller) lines. Around them stitching ground vias were placed to drain the noise that can emerge from these lines due to their higher frequency to ground and not to interfere with the nearby traces. The galvanic isolation is implemented through the two areas that don't have any common traces or copper polygons; their only connection being done with the help of the optocouplers. The signal traces in both areas are routed respecting the standard 50Ω impedance of the line using a width of 0.3mm (calculated using a tool provided by the manufacturer shown in Figure 5.10.2.). The differential pair trace of the RS-485 line, having a characteristic impedance of 120Ω, it has been computed that a trace width of 0.12mm and a gap of 0.2mm is necessary. The traces that carry much current, the power traces, are routed with a width of 0.4mm in order to be able to provide the needed current (see Appendix 9.5.). Also, polygons are used on the 3rd layer in order to provide a small impedance path for the current to reach all the components. For signal integrity reasons, under the antenna of the Wi-Fi module, a cut-out area is performed in order to minimize the influence of the emitted 2.4GHz field on the traces from the board and thus suppress the possibility that the information carried by the signals gets erroneous. Moreover, on the top and bottom layers, a ground polygon is present filling the empty space between traces and pads with the same purpose, as these layers are the most susceptible to external noise influence. At the edge of these polygons, as well as inside them, stitching vias are present to ensure that all around the area is the same ground potential, preventing thus ground bouncing, and ensuring that the return currents don't spread in the reference plane. The traces are also placed at a distance from each other to reduce the capacitive coupling and the small distance to the reference plane reduces the inductive one. The bypass capacitors placed at every power pin on the board drain the noise component of the signal to ground, realizing also a decoupling function by providing a local energy reservoir for the corresponding pin. The decoupling capacitor for the Wi-Fi module feeds a polygon that supplies the module and the adjacent components. All the vias used have a hole diameter of 0.3mm and a pad diameter of 0.7mm, providing a current capability of 2.64A, a very low capacitance of 2.8pF, an inductance of 1.3nH and a DC resistance of 0.9mΩ (see Appendix 9.4.), being thus suitable to be used for transitions of power traces as well as of the signal ones.

Impedance (Ω)	Type	Signal Layer	Top Ref	Bottom Ref	Trace Width	Trace Spacing	Impedance trace to copper
50	Coplanar Single Ended	L1	/	L2	0.3424	/	0.5080
120	Coplanar Differential Pair	L1	/	L2	0.1168	0.2032	0.2032

Figure 5.10.2. Traces' characteristics according to the target impedance

Figure 5.10.3. illustrates a power simulation done in Altium with the Power Analyzer extension by Keysight. In the left side there can be observed how the voltage drop of the 3.3V power polygon occurs on the 3rd layer, the worst case getting to 3.29V, an acceptable one. The same is applied for the 5V polygon (right figure), reaching a voltage drop down to 4.98V.

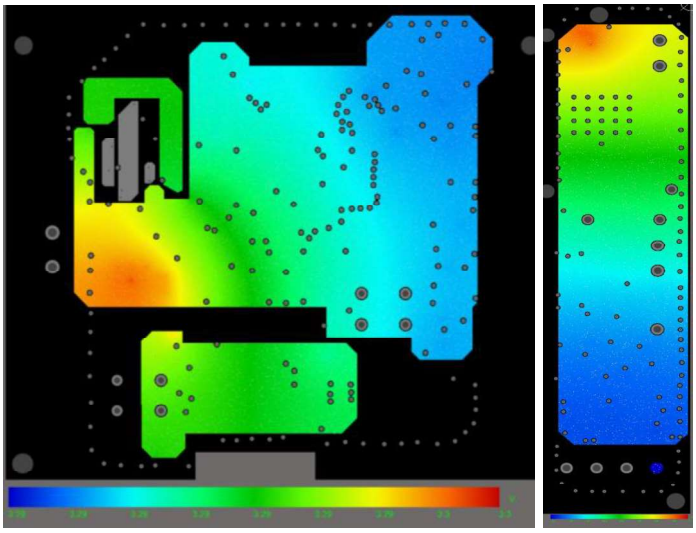

Figure 5.10.3. Power analysis for 3.3V supply (left) and 5V supply (right)

5.11 Software implementation

In figure 5.11.1., the logic scheme of the software implementation is shown. The parameters are measured iteratively using the methods previously presented and then sent via UART0 to the Wi-Fi module which is programmed using Arduino IDE. This communication is performed by handshake protocol: the module waits till the Wi_Fi_T_CTR signal gets high, reads then the data from the serial port after which this signal is pulled low by ATmega64 and puts Wi-Fi_R_INTR high for 2ms to signalize that it has successfully fetched the data. After every fetched data, this is transmitted using the MQTT protocol to ThingsBoard. First, a PubSubClient object (mqtt) is instantiated which will be used for publishing and subscribing messages with the server. After that, the connection to a 2.4GHz Wi-Fi has to be done using the "ESP8266WiFi.h" library and the ssid and password of the wireless network. The connection to the server having the host name "demo.thingsboard.io" (tbHost) is established using the instantiated object and the specific port for MQTT communication 1883 (mqtt.setServer(tbHost.c_str(), 1883)) [60]. This allows the object to publish messages to the server. Also, by subscribing to RPC requests using the token specific for the device in ThingsBoard, the module can get messages from the server. RPC means that the server executes a method in a different "server", sending a request to the device to execute it. These messages are formed of a topic specifying where the data is published or if it is a request/response in case of modifying control elements and a message (payload) formatted as JSON strings, composed of the name of the measured quantity and its value or the modified value of the control element; for example, "Air temperature", "21.5". In consequence, a serialization into JSON of the data that has to be transmitted needs to be accomplished and then published, as well as a deserialization when a message is sent from the server to the device. The only possible messages that could be sent from the server refer to the desired air temperature and soil humidity (in form of

knobs) that can be controlled by the user, the control of the relay (a switch) and the enabling or disabling of the fuzzy control system (through the "AUTO" button), along with a switch to manually control the state of the water pump. Publishing messages to the server is done iteratively, whenever data is coming from ATmega64. Considering though the high power consumption of this operation, sending via Wi-Fi is performed only with a period that correspond to five measurement iterations performed by ATmega64.

For proper functioning of the ATmega64, first the fuse bits need to be set. This way the microcontroller is taking as clock source the external oscillator (low fuse is 0xEF), has enabled the JTAG (Joint Test Action Group) interface and the ICSP (high fuse 0x99), and also ATmega103 compatibility mode is disabled (extended fuse 0xFF). As such, ATmega64 performs the measurements of the parameters iteratively, computing the duty cycle for the water pump control only if the water level is right and the control system is enabled. A few interrupts are also implemented: Timer0 interrupt for generating the PWM signal for the water pump control, external interrupt triggered by the water level sensor (active on rising edge to signalize full level and falling edge for empty), Timer2 interrupt to collect the samples with the ADC and RX1 interrupt when ESP-12E sends a byte to it to transmit the message gotten from the server.

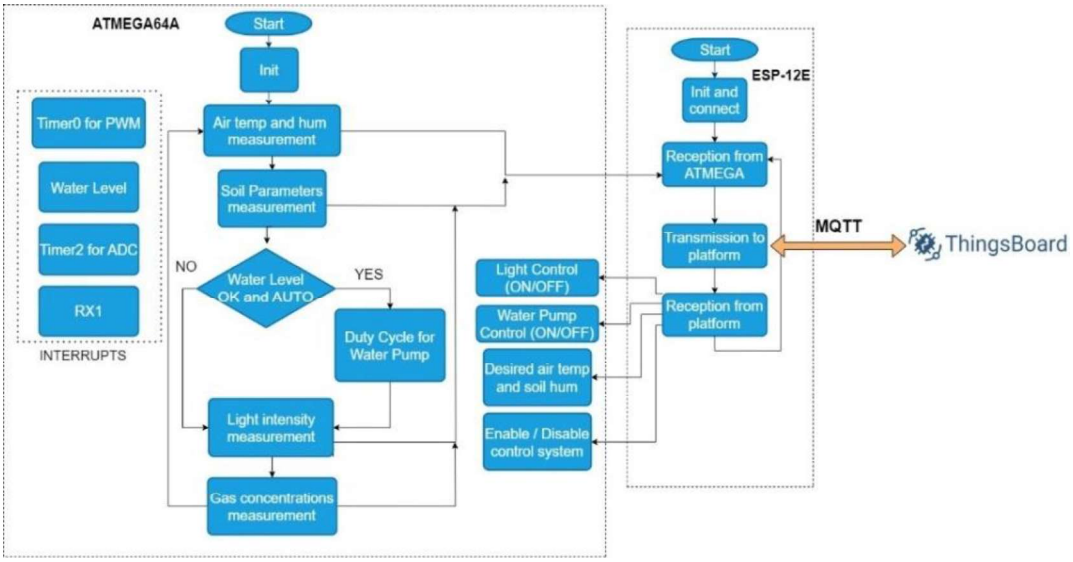

Figure 5.11.1. Software logic scheme

Table 5.11.1. shows the correspondence between the byte that is sent by the Wi-Fi module when receiving an RPC from the platform and how this byte is interpreted by ATmega64. This codification is performed in order to simplify the communication, as this is done via an interrupt of the serial port. In this way, the microcontroller only has to read one byte, instead of reading multiple bytes as it would have been the case for transmitting the actual values for the desired temperature and humidity.

Table 5.11.1. Correspondence between the byte sent and the command for ATmega64

Crt. No.	*Byte*	*Action in ThingsBoard*	*Meaning for ATmega64*
1	0 / 1	LIGHT CONTROL switch OFF / ON	Write "0" / "1" on LIGHT_CTR
2	2 / 3	AUTO button ON / OFF	Execute / bypass fuzzy control
3	4 / 5	WATER PUMP switch ON / OFF	Turn ON / OFF Timer 0 with a duty cycle of 50%
4	6…10	Set desired air temperature with the knob	Set desired air temperature (reduced to 5 possible values)
5	11…14	Set desired soil humidity with the knob	Set desired soil humidity (reduced to 4 possible values)

Table 5.11.2. shows how the values chosen by the user for the desired air temperature and soil humidity are interpreted by the microcontroller for the fuzzy control system.

Table 5.11.2. Correspondence between the values chosen and the ones used for the controller

Crt. No.	*Byte*	*Chosen desired temperature / humidity*	*Desired temperature / humidity used for the fuzzy controller*
1	6 / 11	< 20°C / < 30%	12°C / 18%
2	7 / 12	[20°C; 30°C) / [30%; 55%)	25°C / 42%
3	8 / 13	[30°C; 40°C) / [55%; 70%)	35°C / 62%
4	9 / 14	[40°C; 55°C) / ≥ 70%	48°C / 80%
5	10	≥ 55°C	77°C

5.12 LCD

Figure 5.12.1. shows the connections for powering the LCD, applying to VEE (that controls the contrast of the text on the display) a voltage of 0.88V through the voltage divider R_6-R_7 and 3.3V to the supply pin VCC. The control signals EN (enable), RS (register select), RW (read / write) have to be used as followed: for sending a command to the LCD (initialization, clear display, place cursor, move to second line, move at the beginning of the line, turn off cursor), the code for the corresponding command is sent to the data pins D7-D0, RW and RS are pulled low, and EN is given a pulse of 20ms. To write text on the display, the ASCII code of the characters needs to be placed one a time on the data pins, pulling RW low, RS high and sending a pulse to EN. Before performing any of these operations, it has to be checked if the LCD is available to get data by reading the busy flag (D7). This is accomplished by pulling RS low and RW high, signalizing a reading operation from the LCD, and sending a pulse on EN. A value of "0" means the LCD is ready to accept data [61].

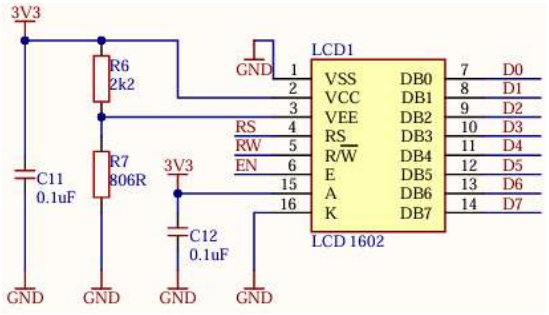

Figure 5.12.1. Circuit for LCD

The code for the commands used throughout the whole programming code of the microcontroller are: 0x38 for initializing, 0x01 for clear display, 0x0E for placing the cursor, 0x0C to turn off the cursor, 0x80 to start from the beginning of the first line, 0xC0 to move to the second line.

6 Experimental Results

Figure 6.1. shows the PCB with its components (left): 1 – LCD, 2 – MICS-6814, 3 – ATmega64A, 4 – I2C sensors (HS3002, BH1721), 5 – ESP-12E, 6 – RS-485 transceiver circuit, 7 – circuit for water level sensor, 8 – water pump control circuit, 9 – light control circuit, 10 – 5V supply (LDO and battery header), 11 – 3.3V supply (LDO and battery header). In the right figure there are the external components: 12 – soil quality sensor, 13 – water pump, 14 – water level sensor. All the signals presented in what follows are measured on the board with ADALM2000 module, having a maximum sample rate of 100MSPS and Scopy – v1.4.1 for software oscilloscope and signal analysis, developed by Analog Devices.

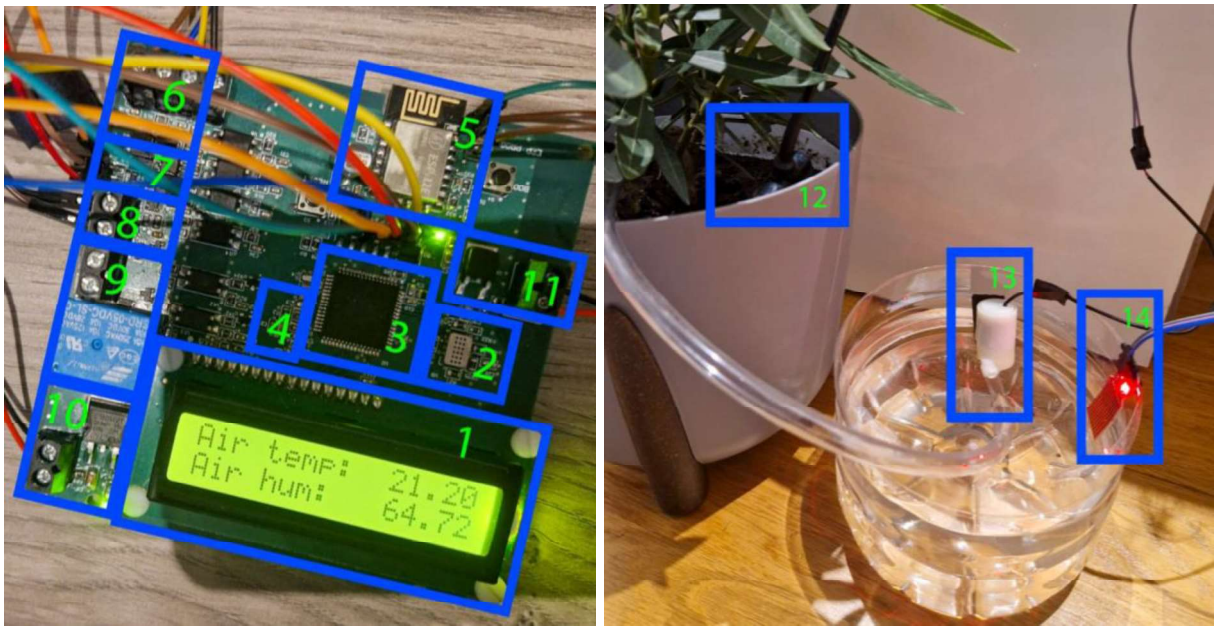

Figure 6.1. PCB (left) and external components (right)

6.1 Reading the fuse bits and programming

For ensuring that the fuse bits of ATmega64 are correctly programmed for proper functioning of the microcontroller, one has to read them, according to Figures 6.1.1. and 6.1.2. (left). It is illustrated how to use the command window for checking these bits by using avrdude, an utility program for AVR microcontrollers and usbasp. The values of these bits are according to the ones stated in chapter 5.11. The sequence for programming ATmega64 is shown in Figure 6.1.2. (right), resulting in a total number of 8228 bytes for the programming file.

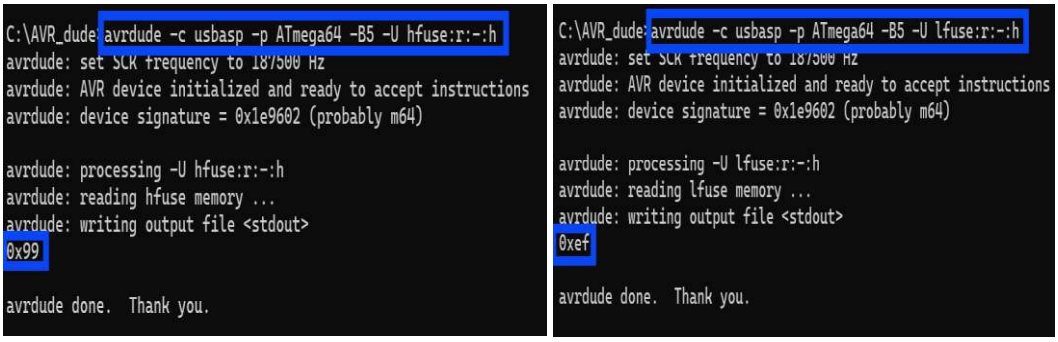

Figure 6.1.1. Reading the high fuse bits (left) and the low ones (right)

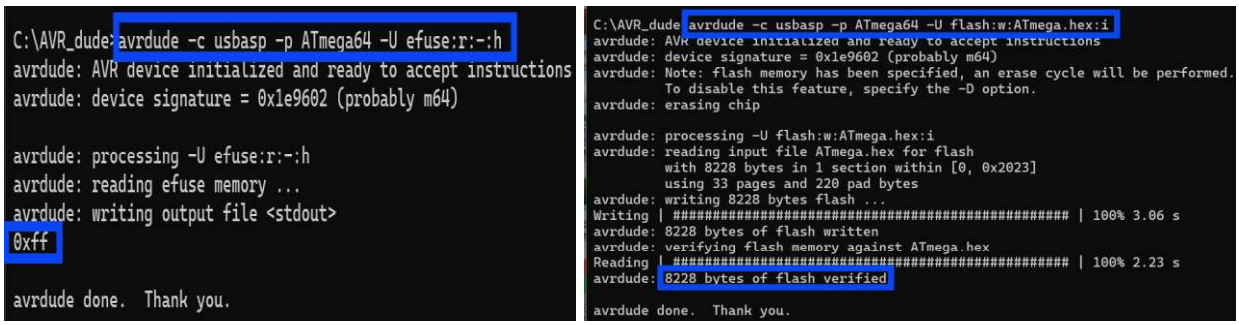

Figure 6.1.2. Reading the extended fuse bits (left) and programming sequence (right)

6.2 I2C communication

6.2.1 HS3002

A signal analysis using the I2C interpreter from Scopy is depicted in Figures 6.2.1. and 6.2.2. The SDA and SCL signals can be observed, as well as the interpretation for the I2C protocol, such as “S” for the start bit, “A” for acknowledge bit, “N” for not acknowledge bit and “P” for stop bit. In the first figure, the sensor’s addressing is represented, with its address to which to write bit is appended, as stated in chapter 5.6. In the next one, the address with the read bit is illustrated, followed by the data bytes. The first two bits from the first data byte are “00”, stating that the data is valid. Next, there are the data bytes for humidity, which converted to decimal correspond to the value 10582, resulting in a humidity of 64.59%, computed using equation (5.6.1). The ones for temperature correspond to the decimal value 6838, so a temperature of 28.86°C, determined using (5.6.2).

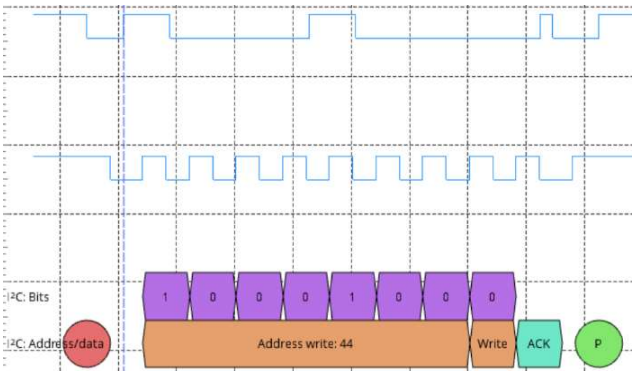

Figure 6.2.1. HS3002 addressing

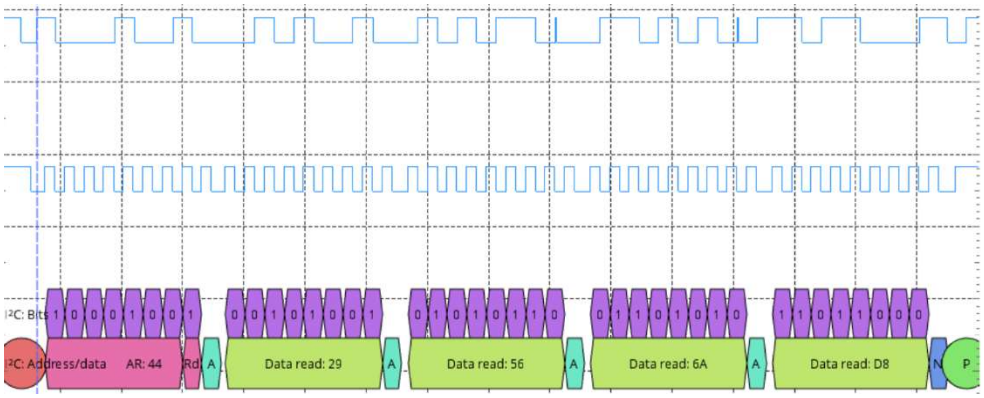

Figure 6.2.2. HS3002 measurement result

6.2.2 BH1721

The below figures show a measurement of the I2C signals for the BH1721 sensor. Figure 6.2.2. (left) show the addressing of the light intensity sensor, together with the initializing of the auto-resolution mode, as specified in chapter 5.6. The request and answer of the measurement result is illustrated in the right figure, the value transmitted by the sensor in decimal being 13, corresponding to a light intensity of 10 lx.

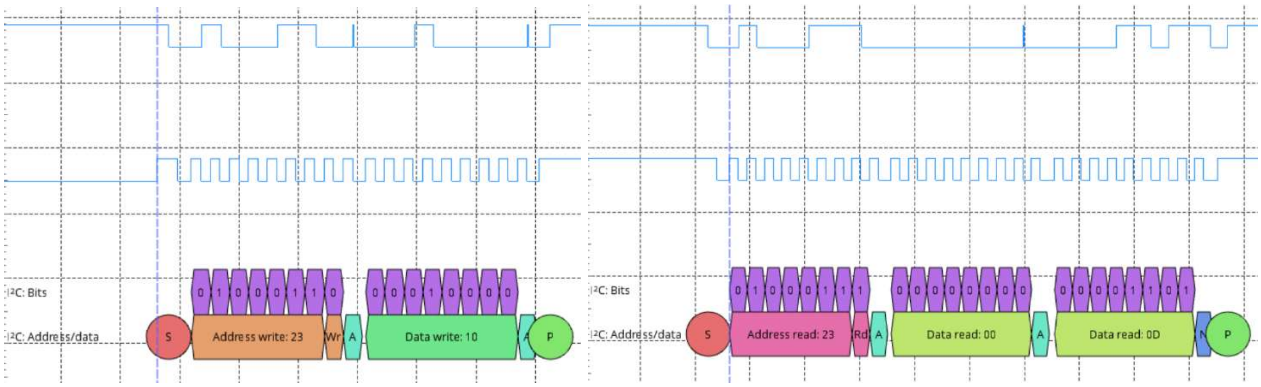

Figure 6.2.2. BH1721 addressing (left) and measurement result (right)

6.3 Optocoupler

Figure 6.3.1. shows the voltage levels measured on the DI trace. The orange waveform illustrates the shape at the input of the optocoupler, so the signal sent by ATmega64, with a voltage of 3.6V for logic “1”, and the purple one represents the shape at the output of the optocoupler, respectively at the DI pin of MAX483, 5V signalizing logic “1” level. There can be observed that these signals are in phase so that the data is interpreted in a correct way, as simulated in Figure 5.4.2. (left).

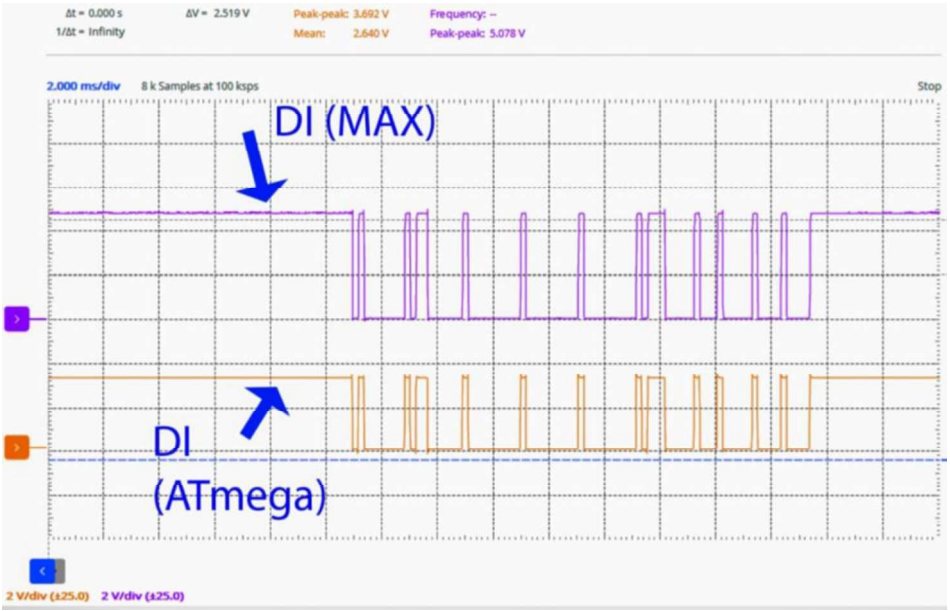

Figure 6.3.1. DI voltage at ATmega64 side (orange) and at the input of MAX483 (purple)

6.4 RS-485 communication

Figure 6.4.1. illustrates the differential signals A and B for transmitting the inquiry frame, as stated in Figure 5.4.3. There can be observed that for representing logic “1”, a difference in voltage of 900mV is present between A and B because of the fail-safe configuration.

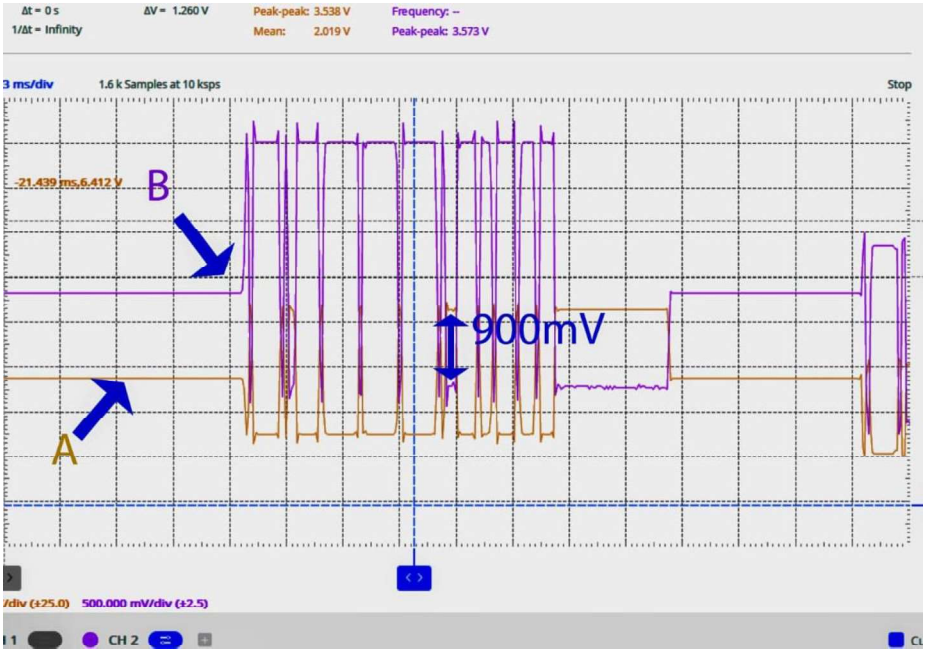

Figure 6.4.1. Inquiry frame

The response frame is presented in Figure 6.4.2., respecting the frame explained in Figure 5.4.4. In both figures there can be observed that for logic “0”, a differential voltage of 3.5V appears between B and A.

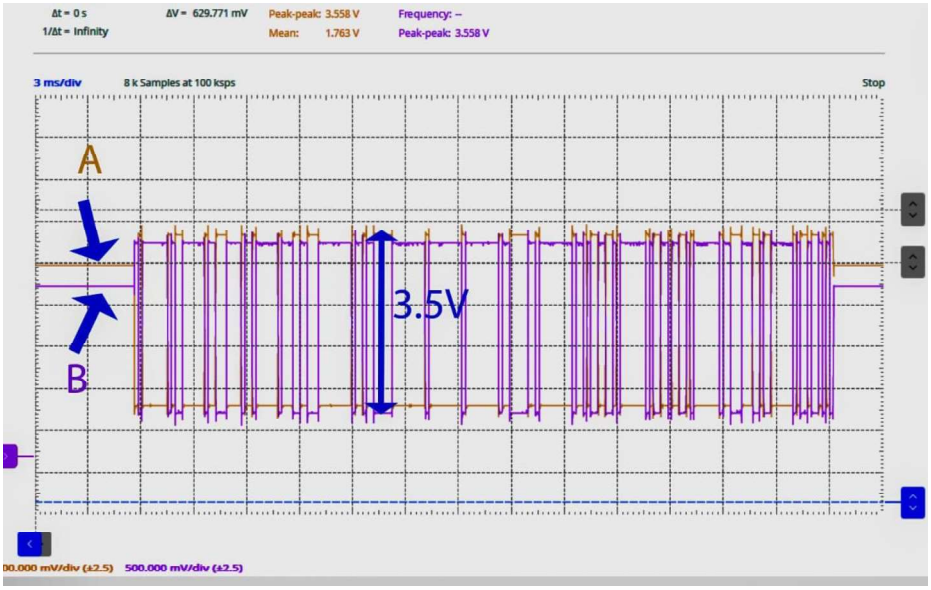

Figure 6.4.2. Response frame

6.5 Hysteresis comparator for water level sensor

A sinusoidal input signal of amplitude 2.5V and offset 2.5V with a low frequency is applied to the hysteresis comparator for testing considerations, generated by ADALM2000 and the output signal of the comparator is measured, plotting the transfer characteristic. This is illustrated in Figure 6.5.1., with the two thresholds: 1.77V the low threshold and 2.23V the high one, close to the simulated values in Figure 5.8.2.

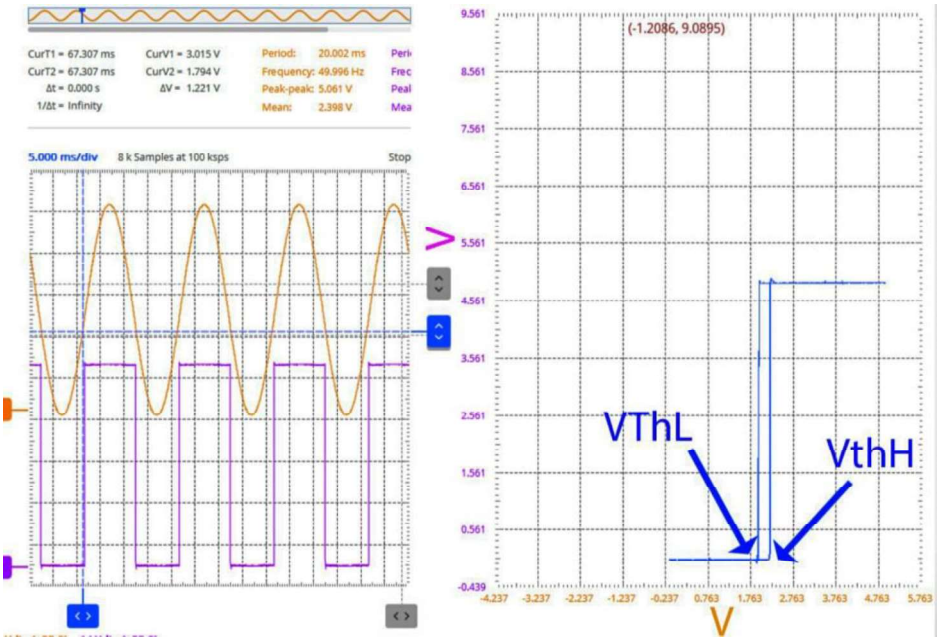

Figure 6.5.1. Input and output signals (left) and transfer characteristics (right)

6.6 Interface in ThingsBoard

Figure 6.6.1. shows the telemetry with the data transmitted by the Wi-Fi module to the device in ThingsBoard. The data is sent by the module in the form of key-value pairs (JSON string), published with the subject “telemetry”.

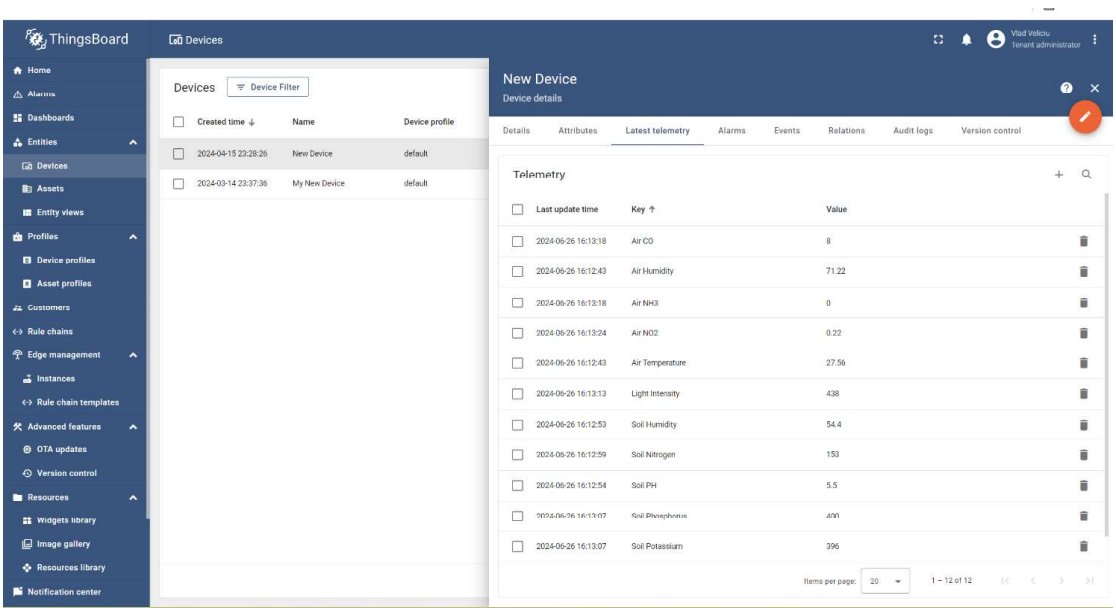

Figure 6.6.1. Telemetry in ThingsBoard

Figure 6.6.2. depicts the user interface with the time charts and the buttons and switch for control. The state of the AUTO button and the sliders are published with the topic “attributes”, the only slider that is automatically changed being the one for the state of the water pump.

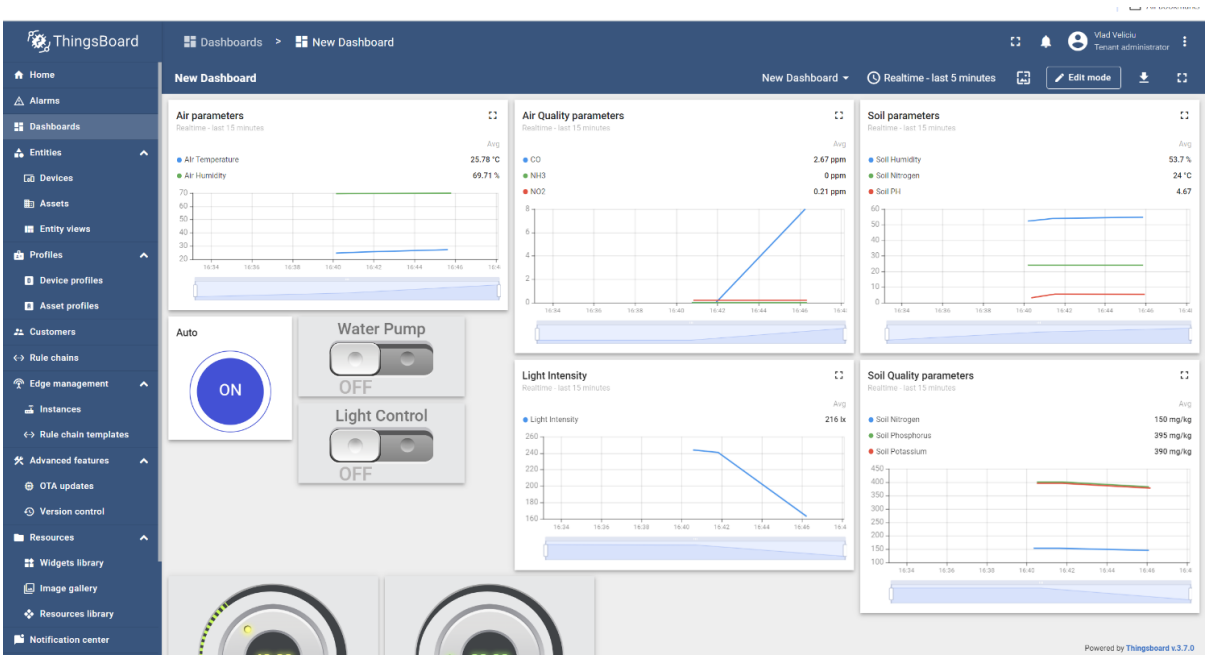

Figure 6.6.2. The data displayed in time series charts

6.7 Testing the control system

Figure 6.7.1. depicts the control elements for the system, in form of buttons, sliders and knobs.

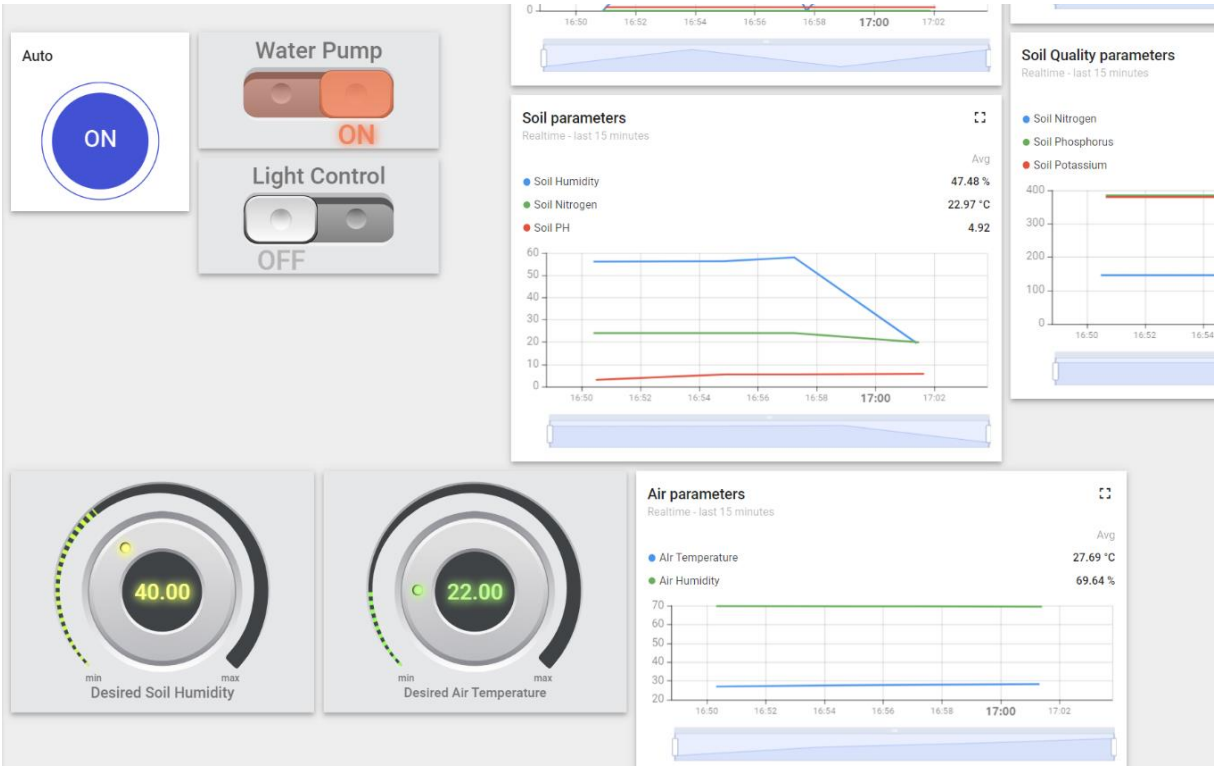

Figure 6.7.1. Control elements in the interface

In figure 6.7.2. illustrates the structure of the JSON string received by the module when the platform performs an RPC call, that is when the user changes the properties of one of the control elements. The figure shows the case when the desired soil humidity is set to 75%. As such, an RPC request is sent to the Wi-Fi module, with the method “setDesiredHum” and parameters the value to be set. The message is deserialized and the corresponding method is executed by the module. This method sets the parameter accordingly and sends the value to ATmega64 via RX interrupt. It responds then with an RPC response by setting the corresponding value in the interface (this is done by publishing the same topic, only replacing “request” with “response”, and serializing the message into JSON string).

```
On message
Topic: v1/devices/me/rpc/request/51
Message: {"method":"setDesiredHum","params":75}
Get info: 75
```

Figure 6.7.2. Topic and message (JSON string) sent by the platform for RPC call

An analysed case is the one for a measured soil humidity of 21.5% and air temperature of 25.19°C, with a desired humidity of 40% and desired temperature 22°C (initial values for the control system). Figure 6.7.2. (left) shows the rule inference for this system, resulting in a duty cycle of 39%. By measuring the signals on the board, illustrated by Figure 6.7.2. (right), a duty cycle of the same value for the command voltage is determined by the system (having the on-time equal with 3.103ms and period of 8.198ms).

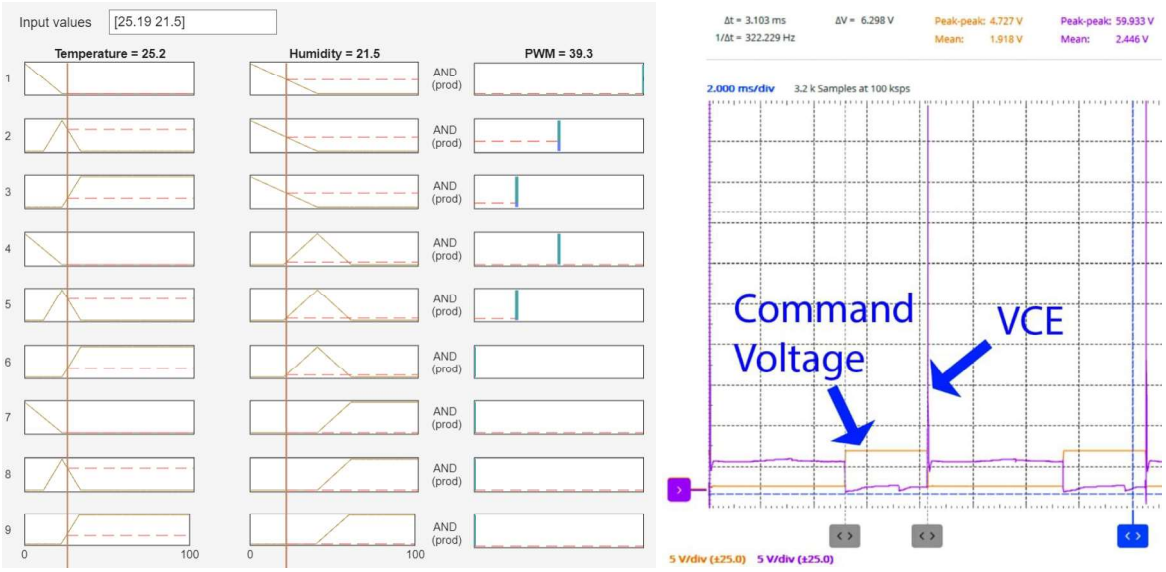

Figure 6.7.2. Rule inference (left), Command voltage and VCE (right)

In Figure 6.7.2. (right) there can be observed that a large overshoot of 50V appears on the transistor at switching from ON state to OFF one. Consequently, a snubber circuit should also be designed to diminish this voltage peak, although the used transistor is able to handle such peaks.

Another analysed case is the one for a measured soil humidity of 73.7% and air temperature of 28.2°C, with chosen desired humidity of 71.95% and desired temperature 43.58°C (interpreted by the control system as 80% and 48°C). Figure 6.7.3. (left) shows the rule inference for this system, resulting in a duty cycle of 55%. By measuring the signals on the board, illustrated by Figure 6.7.3. (right), a duty cycle of 58% is determined by the system (having the on-time equal with 4.8ms and period of 8.227ms).

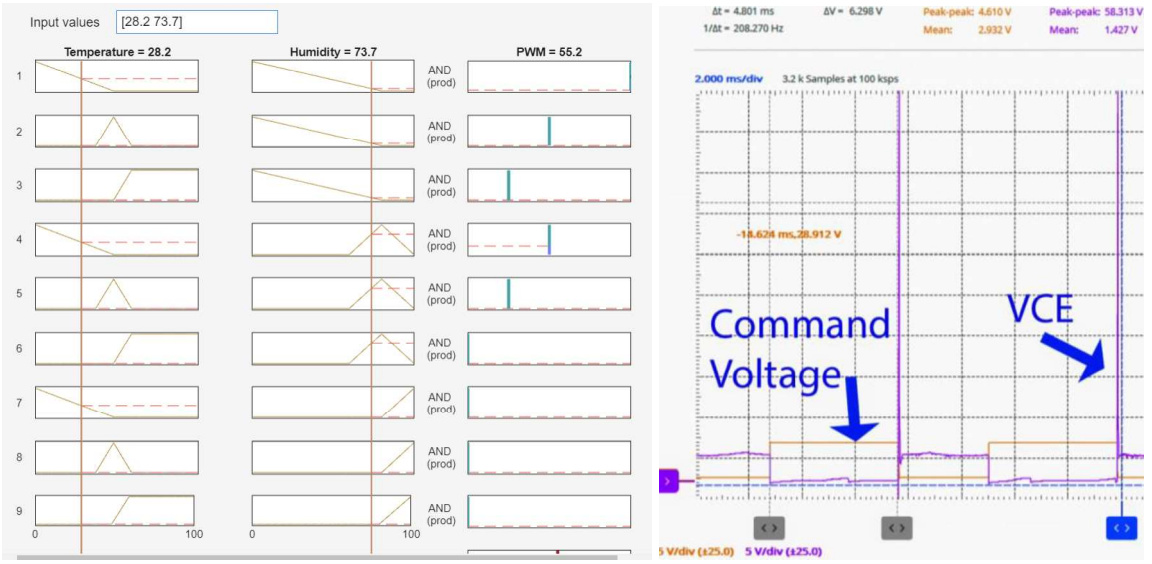

Figure 6.7.3. Rule inference (left), Command signal and VCE (right)

A case where the duty cycle is 100% is presented in Figure 6.7.4. The desired humidity is set to 68.8% and the temperature to 52.81°C, that are interpreted by the system as 62% respectively 62°C. The measured air temperature being 25.19°C and soil humidity 16.2%, the system designed in Matlab will give at the output a duty cycle of 100% (left figure), same as the system implemented on the board (right figure), the command voltage of the transistor being always 5V.

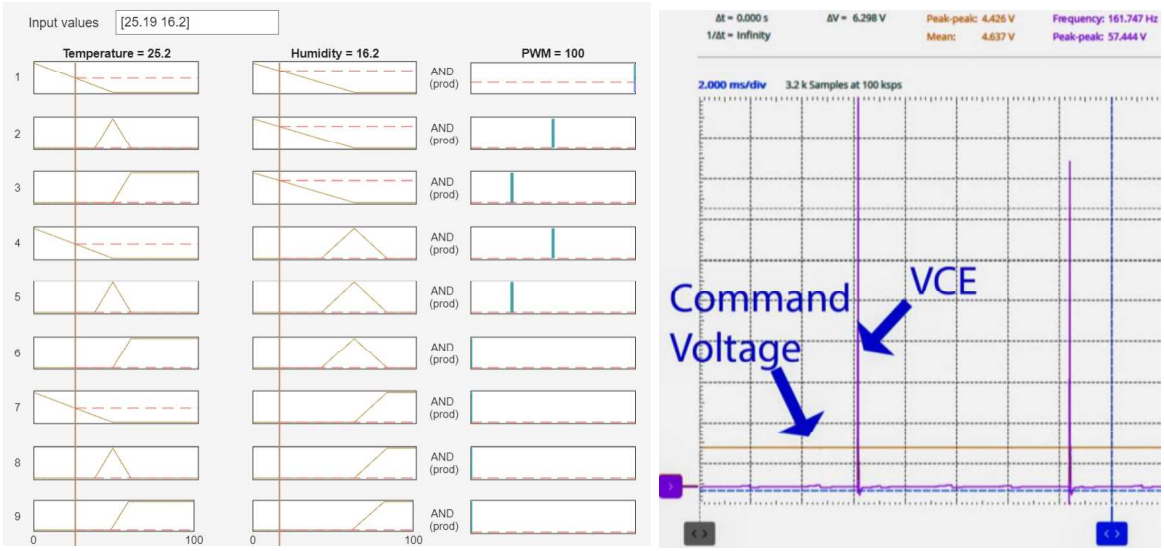

Figure 6.7.4. Rule inference (left), Command signal and VCE (right)

By turning the AUTO button OFF and switching the WATER PUMP slider ON, the pump is turned on with a duty cycle 50%. Figure 6.7.5. shows the evolution of the command signal and the collector-emitter voltage of the transistor, illustrating a duty cycle of exactly 50% (having on time, 4.099ms and a period of 8.198ms).

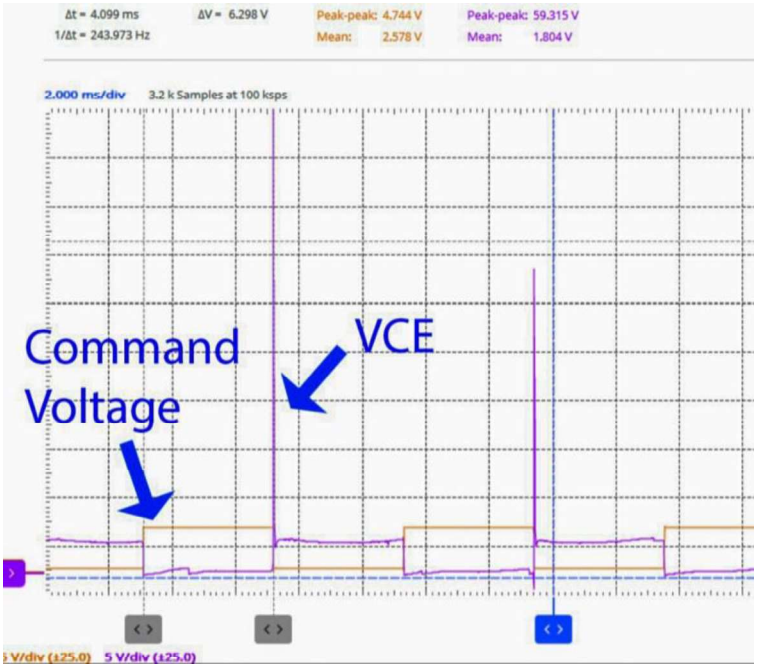

Figure 6.7.5. Command signal and VCE

6.8 MICS-6814

For performing the measurement, the sensing resistance for each type of gas in air has to be determined, as stated in chapter 5.5. The measured resistance in air for CO is 2kΩ and for NO_2 50.25kΩ, values determined with the ADC of the microcontroller, meaning that it also takes into account the resistance of the traces that connect the sensor to the pins. These correspond to concentrations of 8ppm for CO and 0.22ppm for NO_2, which are in the normal parameters for a clean air. The correlation between the sensing resistance, reference resistance in clean air and gas concentration is implemented by the equations (5.5.3) and (5.5.4).

7 Conclusions

The implemented system represents an easy, relatively low-cost solution for an automated greenhouse whose parameters are accessible from everywhere anytime via Wi-Fi. Also, the IoT platform can be used intuitively by the user for visualization and control but also for configuring the interface. The fact that a total number of 12 parameters are determined makes it a complex, useful system for managing a greenhouse. By providing the possibility to the user to control the desired air temperature and soil humidity, so to change the variables of the fuzzy inference control system, it can be used for any type of plant in order to achieve optimal growing conditions. Moreover, the more expensive components from the board are protected from any hazard that can get on the lines that travel a long distance till they reach the PCB through the galvanic isolation. The system is thus more reliable when it comes to maintenance, as the probability of changing the main components which are also of higher cost, is reduced.

Further improvements regarding the system may include a better power management. This means that one single power supply should be used instead of two independent ones, keeping though the galvanic isolation. A possible solution would be designing a flyback converter that ensures this and gives at the output a constant voltage. Moreover, a supply made of solar panels could be put into practice, together with an inverter that would transform the DC signal given at the output of the panels into AC (alternating current), this being fed to the input of the flyback converter. As such, a more reliable and long-lasting power management could be achieved, instead of using batteries that need to be recharged.

Another enhancement could be automating more parameters, not only the water control. For instance, the light control could be implemented in the microcontroller by counting the number of hours when the light intensity is above a certain threshold, meaning daytime. When the intensity is below, the light control circuit could be powered on, ensuring though that the 16 hours of “daytime” are not exceeded.

Considering the arguments presented above, this solution is an adequate one and the improvements would contribute to developing a high-performance system ready to be implemented on a large scale.

8 References

[1] Aznar-Sánchez JA, Velasco-Muñoz JF, López-Felices B, Román-Sánchez IM. An Analysis of Global Research Trends on Greenhouse Technology: Towards a Sustainable Agriculture. Int J Environ Res Public Health. 2020 Jan 20

[2] Xinfa Wang, Vladislav Zubko, Viktor Onychko, Zhenwei Wu and Mingfu Zhao, Research on intelligent building greenhouse plant factory and "3-Positions and 1-Entity" development mode

[3] Gao Junxiang, Du Haiqing, Design of Greenhouse Surveillance System Based on Embedded Web Server Technology, Procedia Engineering, Volume 23, 2011, Pages 374-379, ISSN 1877-7058.

[4] Karanisa, T., Achour, Y., Ouammi, A. et al. Smart greenhouses as the path towards precision agriculture in the food-energy and water nexus: case study of Qatar. Environ Syst Decis 42, 521–546 (2022).

[5] Sebastian-Camilo Vanegas-Ayala, Julio Barón-Velandia, Daniel-David Leal-Lara, "A Systematic Review of Greenhouse Humidity Prediction and Control Models Using Fuzzy Inference Systems", Advances in Human-Computer Interaction, vol. 2022.

[6] P. V. Vimal and K. S. Shivaprakasha, "IOT based greenhouse environment monitoring and controlling system using Arduino platform," 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kerala, India, 2017

[7] Give Your Plants the Best Suitable Environment for Green House. https://www.softwebsolutions.com/resources/greenhouse-environment-control-system.html

[8] How Do Temperature Sensors Work? | Atlas Scientific. https://atlas-scientific.com/blog/how-do-temperature-sensors-work/

[9] The Capacitive Humidity Sensor. https://www.rotronic.com/en-us/humidity_measurement-feuchtemessung-mesure_de_l_humidite/capacitive-sensors-technical-notes-mr

[10] Humidity Sensor - Types and Working Principle - ElectronicsHub USA. https://www.electronicshub.org/humidity-sensor-types-working-principle/#Thermal_Conductivity_Humidity_Sensors.

[11] A. Alfaifi, A. Zaman, and A. Alsolami, "MEMS Humidity Sensors", Humidity Sensors - Types and Applications. IntechOpen, Jan. 04, 2023. doi: 10.5772/intechopen.98361.

[12] Britannica, The Editors of Encyclopaedia. "photoelectric effect". Encyclopedia Britannica, 12 Apr. 2024, https://www.britannica.com/science/photoelectric-effect

[13] Light Sensor | Analog Devices. https://www.analog.com/en/resources/glossary/light-sensor.html.

[14] Photodiode Characteristics and Applications. https://web.mit.edu/6.101/www/reference/Photodiode_Characteristics.pdf

[15] Joseph Wu, "A Basic Guide to I2C", 2022. https://www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=1718166569697&ref_url=https%253A%252F%252Fwww.google.com%252F

[16] Rajan Arora, "I2C Bus Pullup Resistor Calculation", 2015. https://www.ti.com/lit/an/slva689/slva689.pdf?ts=1718174806771&ref_url=https%253A%252F%252Fwww.google.com%252F

[17] Nandagawali, Swati Narsing. Parameters of Soil Fertility (As a Part of Project on Soil Parameters Monitoring With Automatic Irrigation System). no. 4, 2015. https://www.researchpublish.com/upload/book/Parameters%20of%20Soil%20Fertility-2560.pdf

[18] Renkeer, "5 Types Of Soil Sensors - Which Is Best For You?", 2024. https://www.renkeer.com/5-types-soil-sensors/

[19] Mark Harris, "Serial Communications Protocols - Part Four: RS-485 and Baud Rates" | Altium, 2021. https://resources.altium.com/p/serial-communications-protocols-rs-485
[20] RS-485 Interfaces, AN1991, 2018. https://www.renesas.com/us/en/document/apn/an1991-isolating-rs-485-interfaces-high-speed-digital-optocouplers
[21] Air Quality Standards – Tulsa Air Quality. https://tulsaairquality.com/about/air-quality-standards/
[22] What Is a Gas Sensor: Working and Types. https://wiki.dfrobot.com/What_is_a_Gas_Sensor
[23] Types of Relays - Their Construction, Operation & Applications. https://www.electricaltechnology.org/2018/12/what-is-relay-different-types-of-relays-its-operation-applications.html
[24] How and Why to Convert Analog Signals to PWM Signals. https://resources.pcb.cadence.com/blog/2021-how-and-why-to-convert-analog-signals-to-pwm-signals
[25] PWM Driver with BJT Transistor - eMariete. https://emariete.com/en/driver-pwm-transistor-bjt/
[26] Arduino Transistor DC Motor Control, 2016. https://basicelectronicsengineering.blogspot.com/2016/07/arduino-transistor-dc-motor-control.html
[27] From PID to Fuzzy Control. https://www.mstarlabs.com/control/fuzzypid.html
[28] Kamyar Mehran, "Takagi-Sugeno Fuzzy Modeling for Process Control", 2008. https://www.staff.ncl.ac.uk/damian.giaouris/pdf/IA%20Automation/TS%20FL%20tutorial.pdf
[29] What Is a Water Sensor - Working, Types, & Applications – Robocraze. https://robocraze.com/blogs/post/what-is-a-water-sensor
[30] Comparator with Hysteresis (Schmitt Trigger) Calculator - Engineering Calculators & Tools. https://www.allaboutcircuits.com/tools/hysteresis-comparator-calculator/
[31] Josh Schneider, Ian Smalley. What Is a Microcontroller? | IBM, 2024. https://www.ibm.com/think/topics/microcontroller
[32] Microcontroller ISP Programming (Save Money) : 4 Steps (with Pictures) - Instructables. https://www.instructables.com/Microcontroller-ISP-Programming-save-Money/
[33] The Difference And Relationship Between Load Capacitance And External Capacitance Of Crystal Oscillator - Industrial News - News - Zhejiang NeoDen Technology Co., Ltd. 2022. https://www.neodensmt.com/news/load-capacitance-and-external-capacitance-58201664.html
[34] ESP12E WiFi Module : Datasheet, Working & Its Applications, 2023. https://www.watelectronics.com/esp12e-wifi-module/
[35] MQTT Essentials Guide and eBook. https://www.hivemq.com/resources/download-mqtt-ebook/
[36] What Is Remote Procedure Call (RPC)? | Definition from TechTarget. https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC
[37] ThingsBoard — Open-Source IoT (Internet of Things) Platform. https://thingsboard.io/
[38] What Are Lithium-Ion Batteries? | UL Research Institutes, 2021. https://ul.org/research/electrochemical-safety/getting-started-electrochemical-safety/what-are-lithium-ion
[39] How to Calculate the Thermal Resistance of a PCB | Heat Sink Calculator-Blog: Focused on Heat Sink Analysis, Design and Optimization. https://www.heatsinkcalculator.com/blog/how-to-calculate-the-thermal-resistance-of-a-pcb/
[40] AN-2020 Thermal Design By Insight, Not Hindsight, SNVA419C, 2013. https://www.ti.com/lit/an/snva419c/snva419c.pdf

[41] ATmega64A datasheet
[42] ECS-80-10-33-CHN-TR3 datasheet
[43] How to Program ESP8266-12E Using Arduino IDE - Hackster.Io. https://www.hackster.io/sainisagar7294/how-to-program-esp8266-12e-using-arduino-ide-3c892f
[44] MAX483CSA+T datasheet
[45] VO617A datasheet
[46] CWT Soil sensor (NPK type) datasheet
[47] MICS-6814 datasheet
[48] DMG1013T datasheet
[49] HS3002 datasheet
[50] BH1721FVC datasheet
[51] Water Pump datasheet. https://cleste.ro/pompa-de-apa-3-6v.html?gad_source=1&gclid=Cj0KCQjwsuSzBhCLARIsAIcdLm7XSEtjLW96US_j3hz2ur3D-ErAEBwvvVNsipfkTa_Bl3lsJwOyVCYaAqVhEALw_wcB
[52] FMMT495 datasheet
[53] Water level sensor datasheet. https://ardushop.ro/ro/electronica/46-modul-senzor-nivel-apa.html?gad_source=1&gclid=Cj0KCQjwsuSzBhCLARIsAIcdLm5pzxoCZMki7d657TszvOmUQ45jdDE9Km7RfrfV_5kKT32N9c9g1U0aAh5OEALw_wcB
[54] AD8538 datasheet
[55] LD1117DT33 datasheet
[56] Saturn PCB Toolkit - Saturn PCB Design | Saturn PCB Design. https://saturnpcb.com/saturn-pcb-toolkit/
[57] L7805 datasheet
[58] PCB Temperature Calculator. https://www.heatsinkcalculator.com/pcb-temperature-calculator.html
[59] PCB Prototype & PCB Fabrication Manufacturer - JLCPCB. https://jlcpcb.com/
[60] https://www.valvers.com/open-software/arduino/esp32-mqtt-tutorial/#connecting-wifi
[61] LCD1602 datasheet. https://www.waveshare.com/datasheet/LCD_en_PDF/LCD1602.pdf

9 Appendix

9.1 Schematic

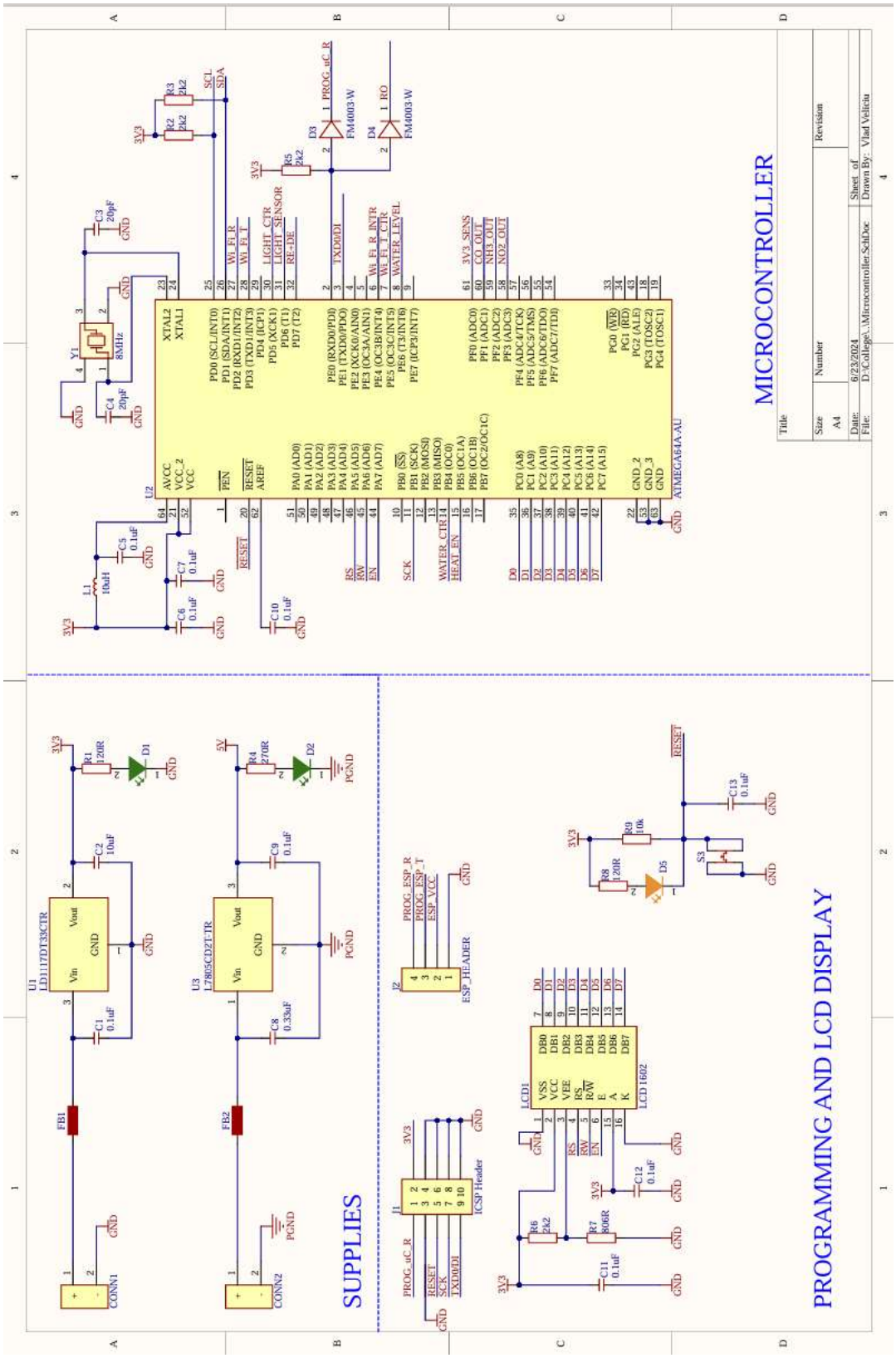

Figure 9.1.1. Schematic for Power supplies, microcontroller, LCD and programming headers

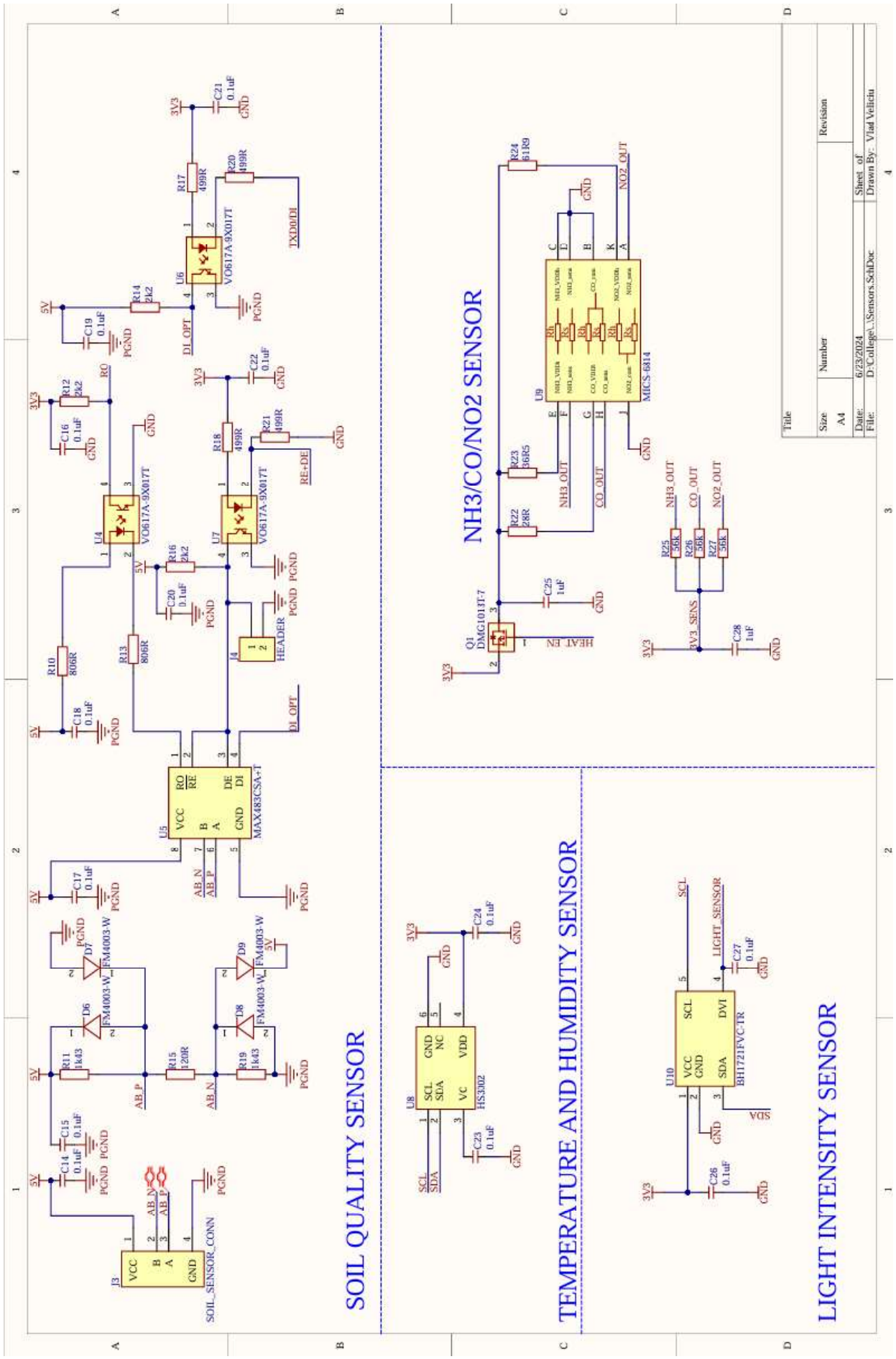

Figure 9.1.2. Schematic for soil quality sensor circuit, I2C sensors, gas concentrations sensor

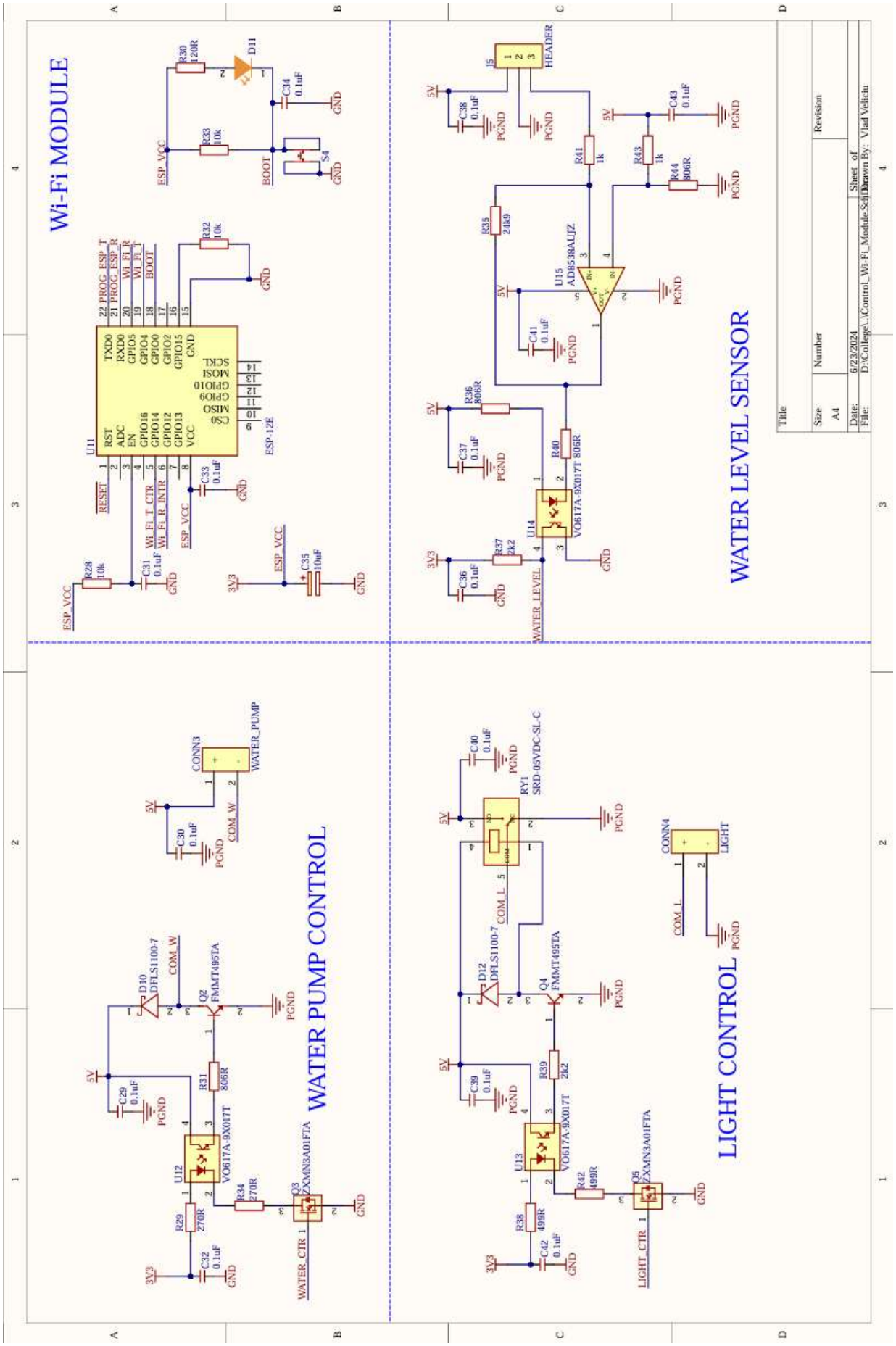

Figure 9.1.3. Schematic for water pump control, light control, Wi-Fi module and water level sensor

9.2 PCB layers

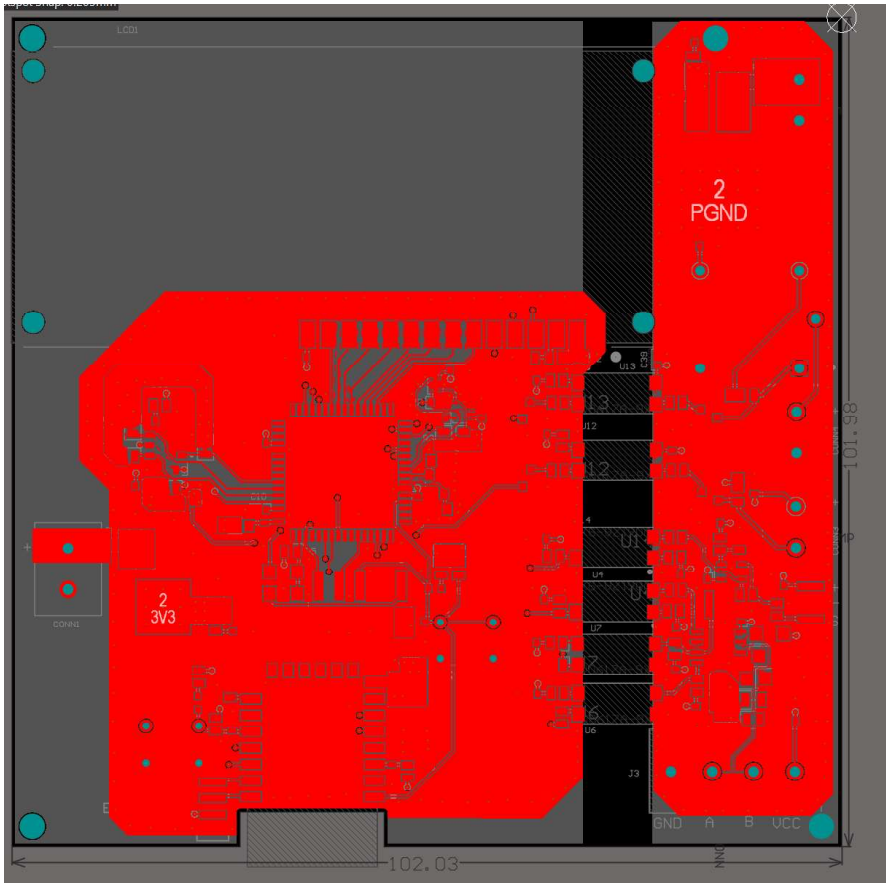

Figure 9.2.1. TOP layer

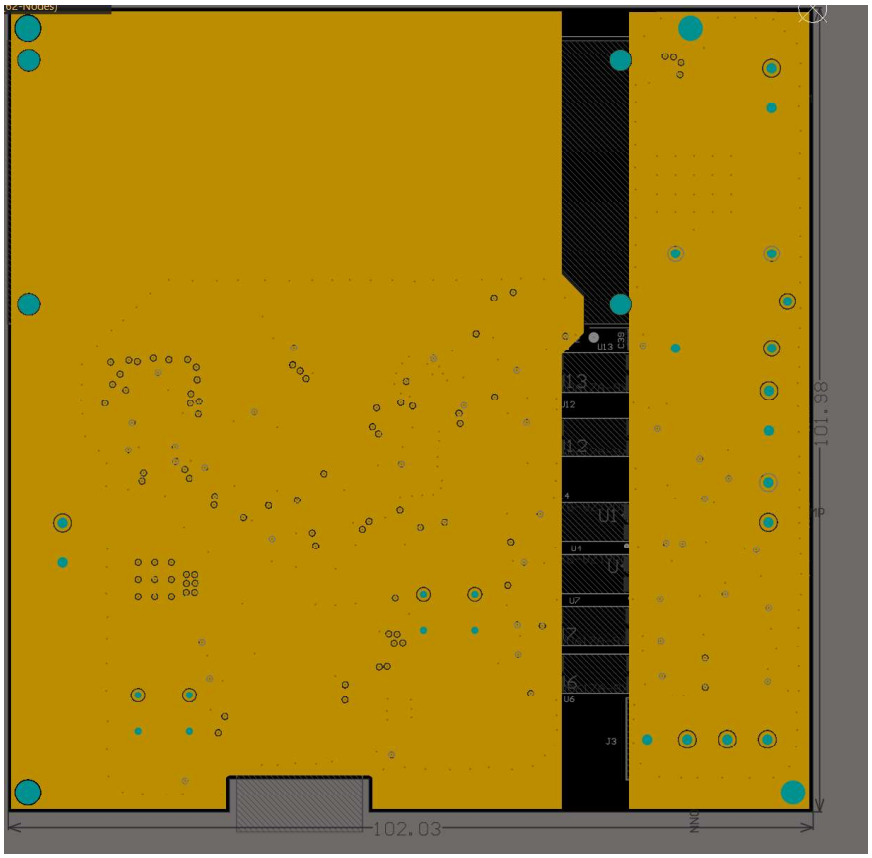

Figure 9.2.2. GND layer

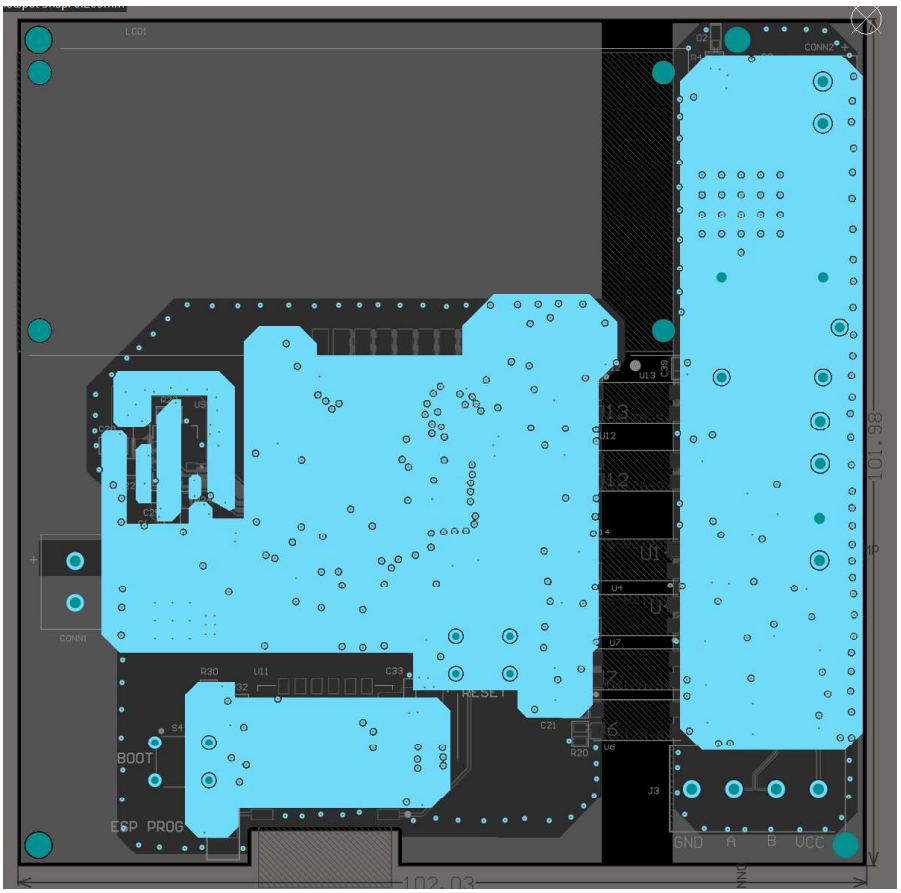

Figure 9.2.3. POWER layer

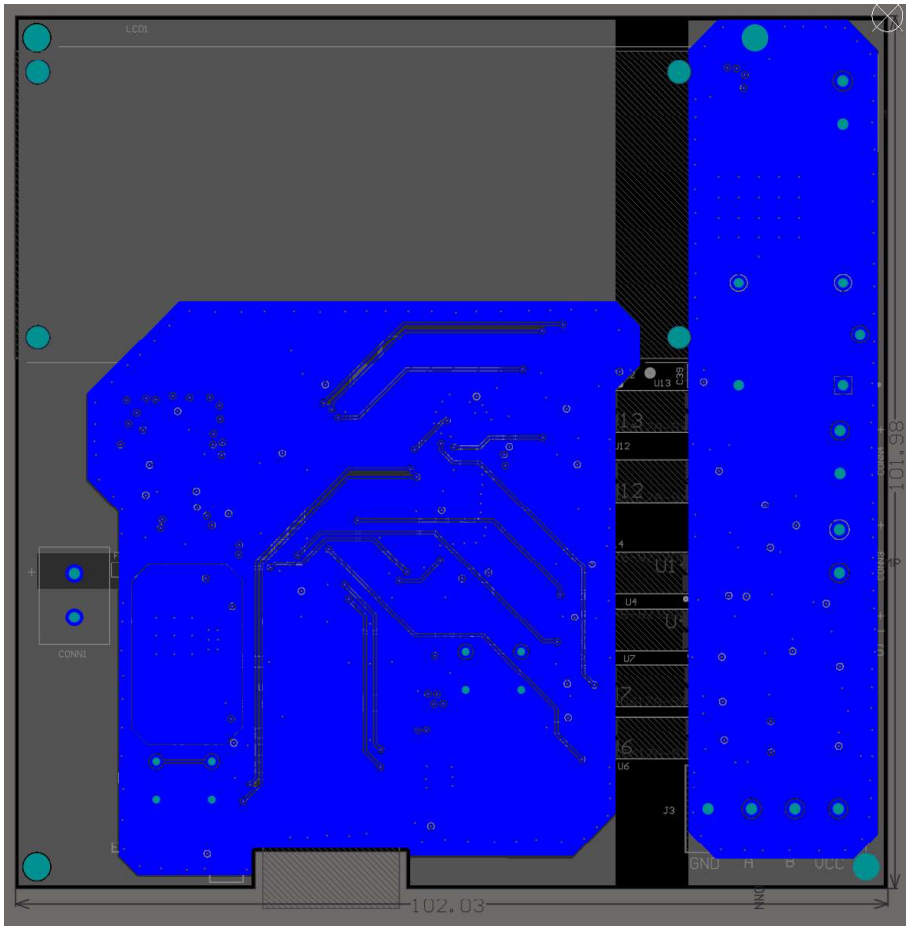

Figure 9.2.4. BOTTOM layer

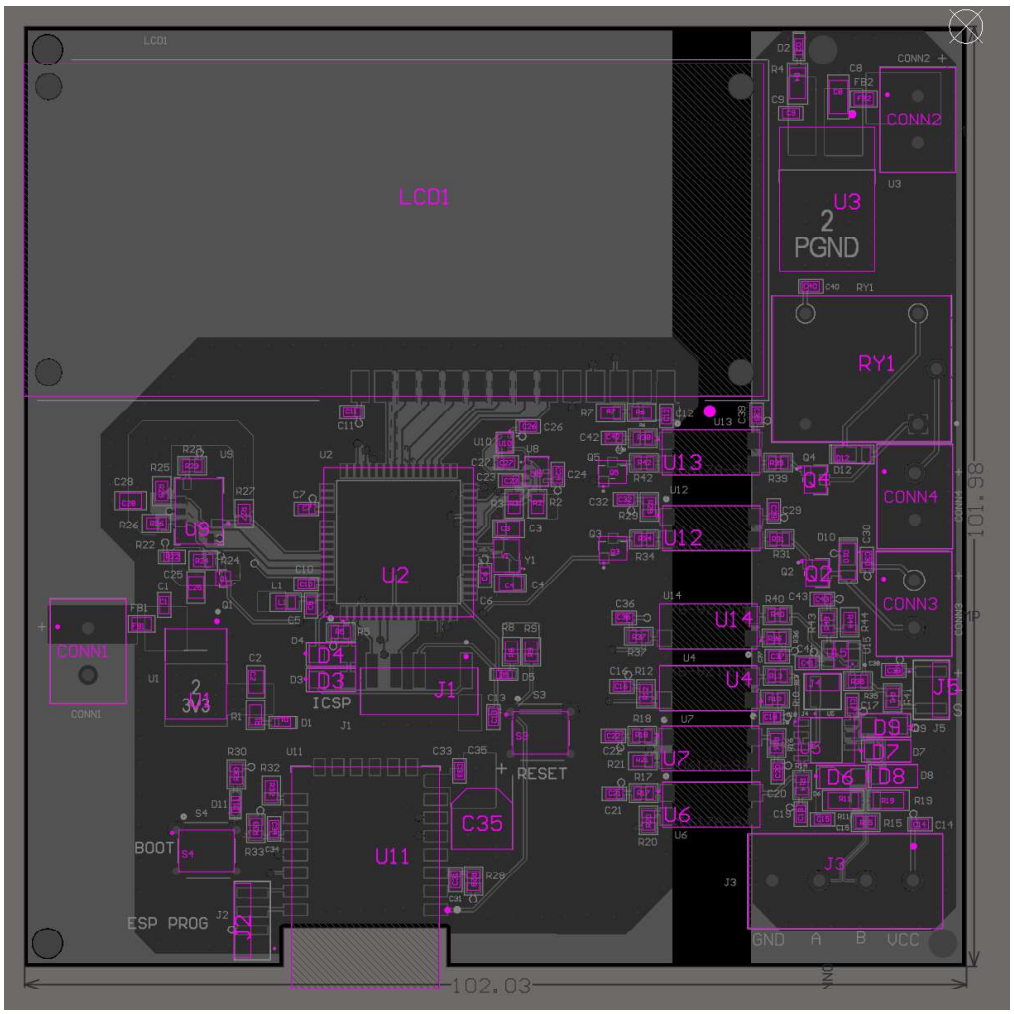

Figure 9.2.5. ASSEMBLY layer

9.3 Bill of materials

Line #	Name	Description	Designator	Quantity	Supplier	Manufacturer	Unit price ($)	Total price ($)
1	0.1uF	Ceramic Capacitor, Multilayer, Ceramic, 10V, 10% +Tol, 10% -Tol, X7R, 15% TC, 0.1uF, Surface Mount, 0603	C1, C5, C6, C7, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22, C23, C24, C26, C27, C29, C30, C31, C32, C33, C34, C36, C37, C38, C39,	36	Digikey	KEMET	0.06	2.16

			C40, C41, C42, C43					
2	10uF	Ceramic Capacitor, Multilayer, Ceramic, 10V, 10% +Tol, 10% -Tol, X5R, 15% TC, 10uF, Surface Mount, 0805	C2	1	Digikey	KEMET	0.2	0.2
3	20pF	Ceramic Capacitor, Multilayer, Ceramic, 25V, 5% +Tol, 5% -Tol, C0G, 30ppm/Cel TC, 0.00002uF, Surface Mount, 0805	C3, C4	2	Digikey	KEMET	0.07	0.14
4	0.33uF	Ceramic Capacitor, Multilayer, Ceramic, 100V, 10% +Tol, 10% -Tol, X7R, 15% TC, 0.33uF, Surface Mount, 1206	C8	1	Digikey	KEMET	0.31	0.31
5	1uF	Ceramic Capacitor, Multilayer, Ceramic, 16V, 5% +Tol, 5% -Tol, X7R, 15% TC, 1uF, Surface Mount, 0805	C25, C28	2	Digikey	KEMET	0.47	0.94
6	10uF	Aluminum Electrolytic Capacitor, Polarized, Aluminum (wet), 50V, 20% +Tol, 20% -Tol, 10uF, Surface Mount, 2626	C35	1	Digikey	KEMET	0.48	0.48

7	BAT_TERM	Strip Terminal Block, 24A, 2.5mm2, 1 Row(s), 1 Deck(s)	CONN1, CONN2	2	Digikey	Weidmuller	0.72	1.44
8	WATER_PUMP	Strip Terminal Block, 24A, 2.5mm2, 1 Row(s), 1 Deck(s)	CONN3	1	Digikey	Weidmuller	0.72	0.72
9	LIGHT	Strip Terminal Block, 24A, 2.5mm2, 1 Row(s), 1 Deck(s)	CONN4	1	Digikey	Weidmuller	0.72	0.72
10	150060VS75000	Single Color LED, Bright Green, Water Clear, 1.1mm	D1, D2	2	Digikey	Wurth Electronics	0.15	0.3
11	FM4003-W	Rectifier Diode, 1 Phase, 1 Element, 1A, 200V V(RRM), Silicon, DO-214AC	D3, D4, D6, D7, D8, D9	6	Digikey	Rectron	0.18	1.08
12	150060AS75000	Single Color LED, Amber, Water Clear, 1.1mm	D5, D11	2	Digikey	Wurth Electronics	0.15	0.3
13	DFLS1100-7	Rectifier Diode, Schottky, 1 Phase, 1 Element, 1A, 100V V(RRM), Silicon	D10, D12	2	Digikey	Diodes	0.34	0.68
14	BLM21SP181SZ1D	Ferrite Chip, 1 Function(s), 2.6A, 2 Pin(s)	FB1, FB2	2	Digikey	Murata	0.21	0.42
15	ICSP Header	Headers & Wire Housings CGrid SMT Bkwy Hdr Twy Hdr Tin 10Ckt T&R	J1	1	Digikey	Molex	1.42	1.42

16	ESP_HEADER	Board Connector, 8 Contact(s), 2 Row(s), Male, Straight, 0.079 inch Pitch, Surface Mount Terminal, Black Insulator	J2	1	Digikey	Samtec	1.42	1.42
17	SOIL_SENSOR_CONN	Barrier Strip Terminal Block, 17.5A, 2.5mm2, 1 Row(s), 1 Deck(s)	J3	1	Digikey	Weidmuller	1.6	1.6
18	HEADER	Board Connector, 8 Contact(s), 2 Row(s), Male, Straight, 0.079 inch Pitch, Surface Mount Terminal, Black Insulator	J4	1	Digikey	Samtec	1.42	1.42
19	HEADER	Board Connector, 8 Contact(s), 2 Row(s), Male, Straight, 0.079 inch Pitch, Surface Mount Terminal, Black Insulator	J5	1	Digikey	Samtec	1.42	1.42
20	10uH	General Purpose Inductor, 10uH, 10%, 1 Element, Ferrite-Core, SMD, 0805	L1	1	Digikey	Abracon	0.11	0.11
21	LCD 1602	Display	LCD1	1	AliExpress		1.26	1.26
22	DMG1013T-7	Small Signal Field-Effect Transistor, 0.46A I(D), 20V, 1-Element, P-Channel, Silicon, Metal-oxide	Q1	1	Digikey	Diodes Zetex	0.24	0.24

		Semiconductor FET						
23	FMMT 495TA	Small Signal Bipolar Transistor, 1A I(C), 150V V(BR)CEO, 1-Element, NPN, Silicon	Q2, Q4	2	Digikey	Diodes Zetex	0.38	0.76
24	ZXMN 3A01F TA	Small Signal Field-Effect Transistor, 1.8A I(D), 30V, 1-Element, N-Channel, Silicon, Metal-oxide Semiconductor FET	Q3, Q5	2	Digikey	Diodes	0.34	0.68
25	120R	Fixed Resistor, Thin Film, 0.125W, 120ohm, 150V, 0.5% +/-Tol, 25ppm/Cel, Surface Mount, 0805	R1, R8, R15, R30	4	Digikey	Yageo	0.1	0.4
26	2k2	Fixed Resistor, Metal Glaze/thick Film, 0.125W, 2200ohm, 150V, 0.5% +/-Tol, 100ppm/Cel, Surface Mount, 0805	R2, R3, R5, R6, R12, R14, R16, R37, R39	9	Digikey	Yageo	0.04	0.36
27	270R	Fixed Resistor, Metal Glaze/thick Film, 0.25W, 270ohm, 200V, 1% +/-Tol, 100ppm/Cel, Surface Mount, 1206	R4	1	Digikey	Yageo	0.11	0.11
28	806R	Fixed Resistor, Thin Film, 0.125W, 806ohm, 150V,	R7, R10, R13, R31, R36, R40, R44	7	Digikey	Yageo	0.16	1.12

		0.1% +/-Tol, 25ppm/Cel, Surface Mount, 0805						
29	10k	Fixed Resistor, Thin Film, 0.125W, 10000ohm, 150V, 0.5% +/-Tol, 25ppm/Cel, Surface Mount, 0805	R9, R28, R32, R33	4	Digikey	Yageo	0.05	0.2
30	1k43	Fixed Resistor, Thin Film, 0.25W, 1430ohm, 200V, 1% +/-Tol, 50ppm/Cel, Surface Mount, 1206	R11, R19	2	Digikey	Yageo	0.1	0.2
31	499R	Fixed Resistor, Thin Film, 0.125W, 499ohm, 150V, 0.5% +/-Tol, 50ppm/Cel, Surface Mount, 0805	R17, R18, R20, R21, R38, R42	6	Digikey	Yageo	0.04	0.24
32	28R	Fixed Resistor, Thin Film, 0.1W, 28ohm, 75V, 0.1% +/-Tol, 25ppm/Cel, Surface Mount, 0603	R22	1	Digikey	Yageo	0.15	0.15
33	36R5	Fixed Resistor, Thin Film, 0.125W, 36.5ohm, 150V, 0.1% +/-Tol, 25ppm/Cel, Surface Mount, 0805	R23	1	Digikey	Yageo	0.17	0.17
34	61R9	Fixed Resistor, Thin Film, 0.125W, 61.9ohm,	R24	1	Digikey	Yageo	0.17	0.17

		150V, 0.1% +/- Tol, 25ppm/Cel, Surface Mount, 0805						
35	56k	Fixed Resistor, Thin Film, 0.125W, 56000ohm, 150V, 0.1% +/- Tol, 25ppm/Cel, Surface Mount, 0805	R25, R26, R27	3	Digikey	Yageo	0.11	0.33
36	270R	Fixed Resistor, Thin Film, 0.125W, 270ohm, 150V, 0.1% +/-Tol, 25ppm/Cel, Surface Mount, 0805	R29, R34	2	Digikey	Yageo	0.04	0.08
37	24k9	Fixed Resistor, Thin Film, 0.125W, 24900ohm, 150V, 0.1% +/- Tol, 25ppm/Cel, Surface Mount, 0805	R35	1	Digikey	Yageo	0.16	0.16
38	1k	Fixed Resistor, Thin Film, 0.125W, 1000ohm, 150V, 0.1% +/- Tol, 25ppm/Cel, Surface Mount, 0805	R41, R43	2	Digikey	Yageo	0.11	0.22
39	SRD-05VDC-SL-C	Releu 5 pini aliniabili pe PCB Voltaj comutare : DC 5V Capacitate sarcina max: 10A 250VAC/125VAC/30VDC/28VDC	RY1	1	Sigmanortec		1.05	1.05
40	Pushbutton	BUTON MINI 6X6X5, 4 PINI	S3, S4	2	Sigmanortec		0.28	0.56

41	LD1117DT33CTR	Fixed Positive LDO Regulator, 3.3V, 1.3V Dropout, BIPolar, PSSO2	U1	1	Digikey	STMicroelectronics	0.44	0.44
42	ATMEGA64A-AU	RISC Microcontroller, 8-Bit, FLASH, AVR RISC CPU, 16MHz, CMOS, PQFP64	U2	1	Digikey	Microchip	5.74	5.74
43	L7805CD2T-TR	Fixed Positive Standard Regulator, 5VBIPolar, PSSO2	U3	1	Digikey	STMicroelectronics	0.73	0.73
44	VO617A-9X017T	Transistor Output Optocoupler, 1-Element, 5300V Isolation	U4, U6, U7, U12, U13, U14	6	Digikey	Vishay	0.26	1.56
45	MAX483CSA+T	Line Transceiver, 1 Func, 1 Driver, 1 Rcvr, CMOS, PDSO8	U5	1	Digikey	Maxim	4.13	4.13
46	HS3002	Temperature Sensor	U8	1	Digikey	IDT	4.19	4.19
47	MICS-6814	CO/NO2/NH3 Triple Gas Sensor Metaloxide Gas Detection Sensor SMD T/R	U9	1	Digikey	Amphenol	11.81	11.81
48	BH1721FVC-TR	Analog Circuit, 1 Func, PDSO5	U10	1	Digikey	Rohm	3.84	3.84
49	ESP-12E	MODUL WIFI ESP8266, ESP12-E, A.I., ORIGINAL	U11	1	Sigmanortec		5.13	5.13
50	AD8538AUJZ	Operational Amplifier, 1 Func, 30uV Offset-Max,	U15	1	Digikey	Analog Devices	1.63	1.63

		CMOS, PDSO5						
51	8MHz	Crystal 8.0000MHZ 10PF SMD	Y1	1	Digikey	ECS International	0.58	0.58
52	Soil Quality Sensor	CWT Soil Sensor (NPK Type)		1	AliExpress	ComWin Top	21.7	21.7
53	Water Pump	Submersible water pump 3-6VDC		1	Sigmanortec		2.17	2.17
54	Water Level Sensor	Water level sensor module 3-5V		1	Sigmanortec		1.08	1.08
TOTAL								90.47

9.4 Via properties

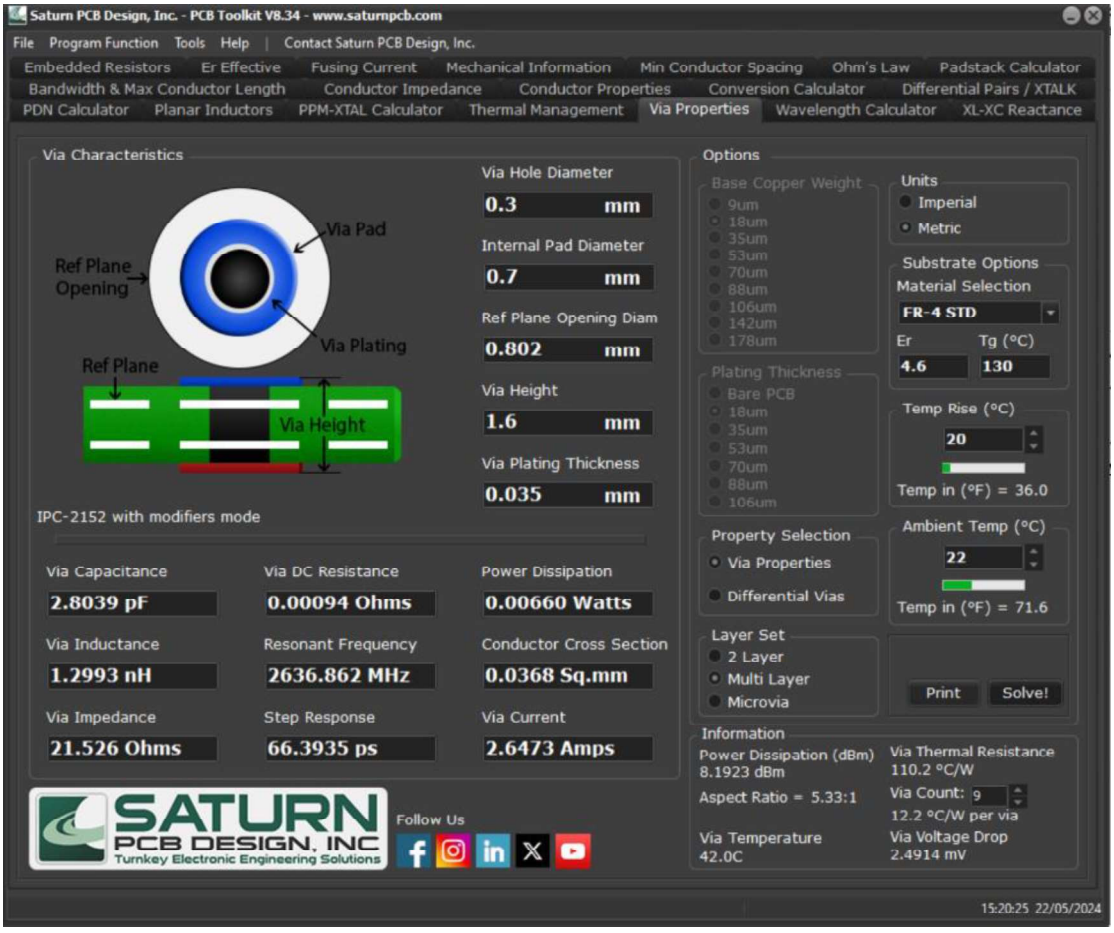

Figure 9.4.1. Via calculator using Saturn

9.5 Properties of the conductors used for power traces

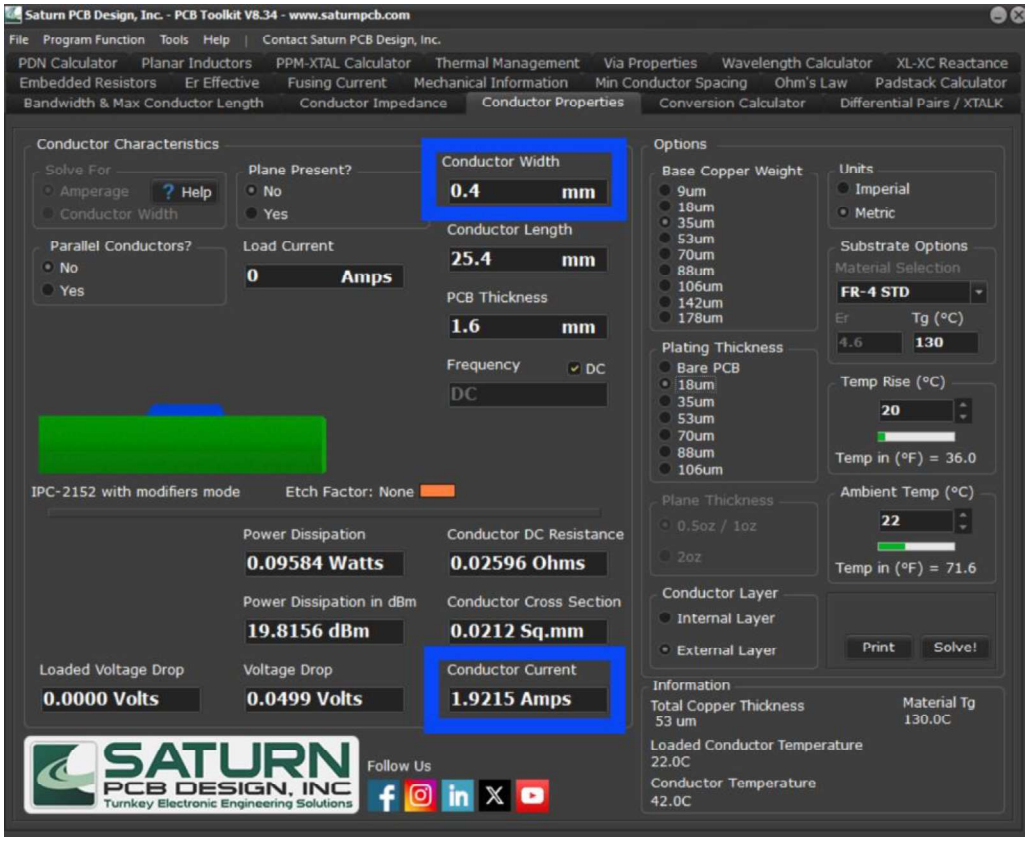

Figure 9.5.1. Saturn calculations

9.6 Junction temperature calculator

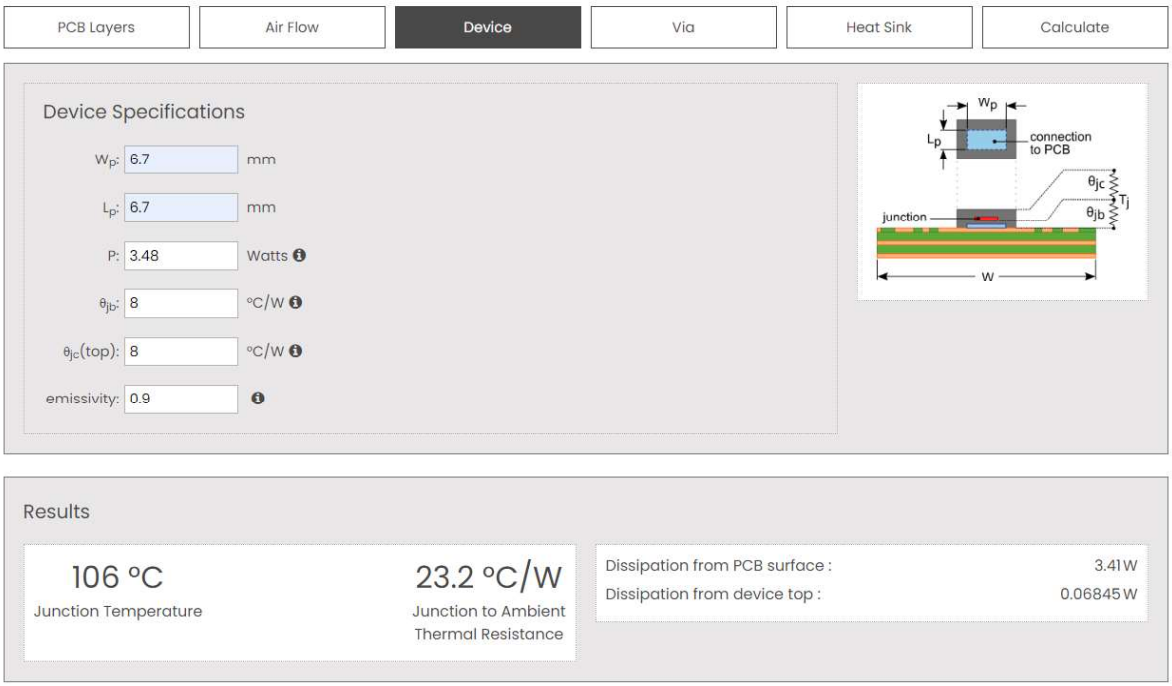

Figure 9.6.1. Junction temperature for 3.3V LDO

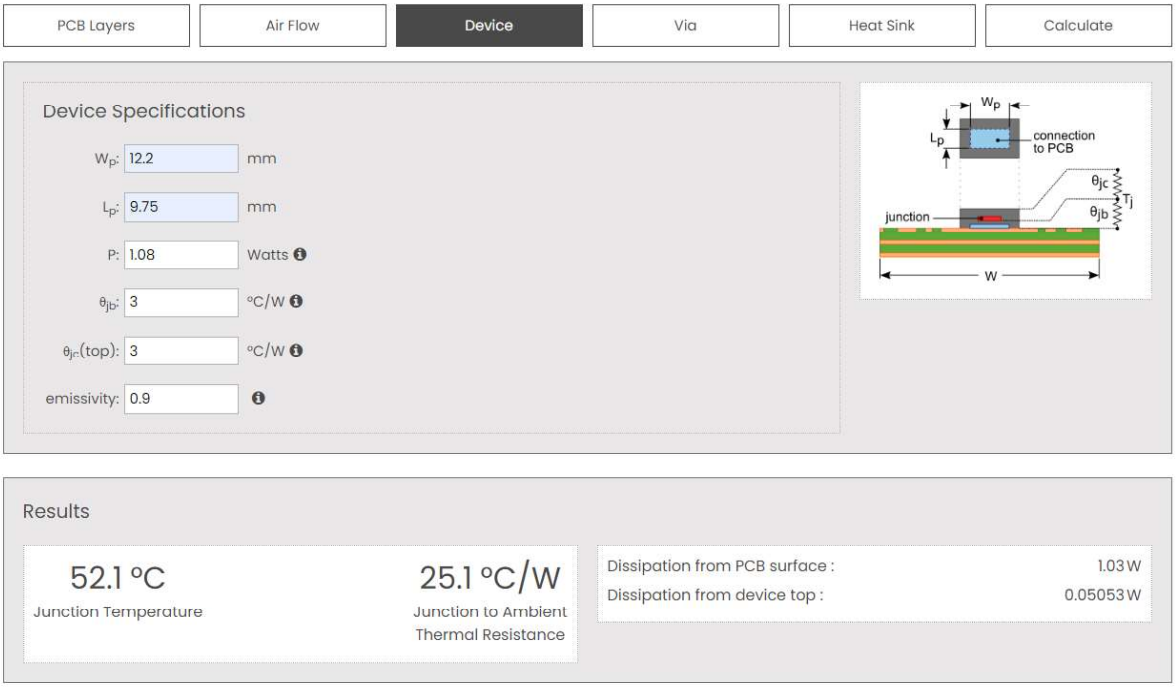

Figure 9.6.2. Junction temperature for 5V LDO

9.7 3D model of the board

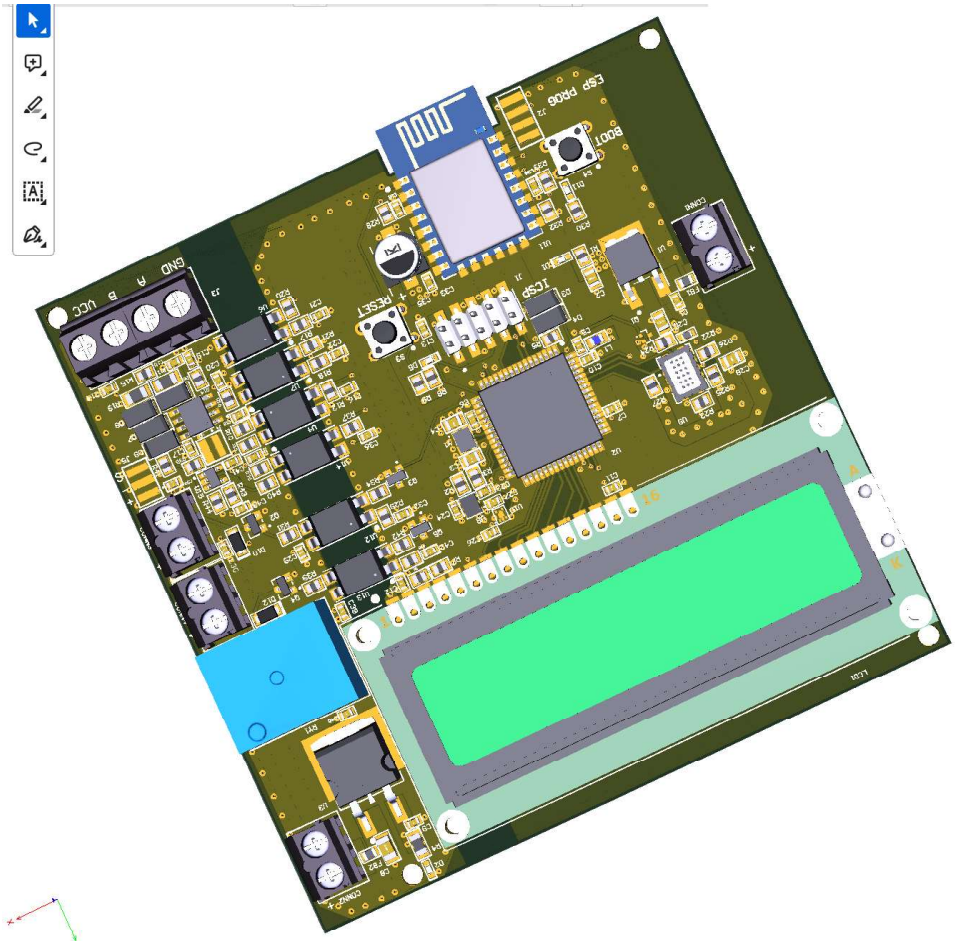

Figure 9.7.1. 3D model

9.8 Code for ATmega64A

LCD.h

```
void LCDready() {

	PORTA = PORTA & 0b01011111
;//write EN 0, RS 0
	PORTA = PORTA | (1<<PINA6)
;//write RW to 1 (want to read from LCD)
	DDRC = DDRC & 0b01111111;//make D7
input
	PORTC = PORTC |
(1<<PINC7);//activate D7 pull-up
	while(PINC7 == 1) {//read busyFlag
		PORTA = PORTA &
0b01111111;//make EN low
		_delay_ms(20);
		PORTA = PORTA |
(1<<PINA7);//make EN high
	}
	PORTA = PORTA & 0b01111111;//make
EN low
}

void LCDcommand(unsigned char command){

	LCDready();
	DDRC = 0xFF;//make output
	PORTC = command;
	PORTA = PORTA & 0b10011111;//make
RW 0 (want to write to LCD), RS 0
	PORTA = PORTA | (1<<PINA7);//make
EN 1
	_delay_ms(20);
	PORTA = PORTA & 0b01111111;//EN=0
}

void LCDwrite(unsigned char data){

	LCDready();
	DDRC = 0xFF;
	PORTC = data;
	PORTA = PORTA & 0b10111111;//make
RW 0 (want to write to LCD)
	PORTA = PORTA | (1<<PINA7) |
(1<<PINA5);//make RS1, EN 1
	_delay_ms(20);
	PORTA = PORTA & 0b01111111;//EN=0
}

void LCDdisplay(unsigned char* message){

	while (*message!=0){
		LCDwrite(*message);
		message++;
	}
}

void displayLCD2Dec(uint16_t x){
	LCDwrite(x/1000%10 + 0x30);
	LCDwrite(x/100%10 + 0x30);
	LCDwrite(0x2E);//comma
	LCDwrite(x/10%10 + 0x30);
	LCDwrite(x%10 + 0x30);
}

void displayLCD1Dec(uint16_t x){
	LCDwrite(x/100%10 + 0x30);
	LCDwrite(x/10%10 + 0x30);
	LCDwrite(0x2E);//comma
	LCDwrite(x%10 + 0x30);
}

void displayLCD0Dec(uint16_t x){

	LCDwrite(x/100%10 + 0x30);
	LCDwrite(x/10%10 + 0x30);
	LCDwrite(x%10 + 0x30);
}

void displayLCD0Dec4(uint16_t x){
	LCDwrite(x/1000%10 + 0x30);
	LCDwrite(x/100%10 + 0x30);
	LCDwrite(x/10%10 + 0x30);
	LCDwrite(x%10 + 0x30);
	LCDwrite(x%10 + 0x30);
}
```

Fuzzy.h

```
uint16_t desiredTemp = 2200;
uint16_t desiredHum = 4000;

const int no_In_Temp = 3;//number of input Temp mf
const int no_In_Hum = 3;//number of input Hum mf
const int no_Outputs = 1;//number of outputs, 1, PWM
const int no_Rules = 9;//number of rules

//coefficients for output PWM
const int pwm[]={0,25,50,100};

//rule base
```

```
/*
1. Temp small and Humidity dry -> PWM 100  3
2. Temp small and Humidity fine -> PWM 50  2
3. Temp small and Humidity wet -> PWM 0    0
4. Temp medium and Humidity dry -> PWM 50  2
5. Temp medium and Humidity fine -> PWM 25 1
6. Temp medium and Humidity wet -> PWM 0   0
7. Temp high and Humidity dry -> PWM 25    1
8. Temp high and Humidity fine -> PWM 0    0
9. Temp high and Humidity wet -> PWM 0     0
*/
const int rules[]={3,2,0,2,1,0,1,0,0};

double F_trimf(uint16_t x, int p[]){//Triangular Membership Function
	int a=p[0], b=p[1], c=p[2];
	double degree=0;
	if ((x > a) && (x < c)) {
		if (x <= b)
			degree=(double)(x-a)/(double)(b-a);
		else
			degree=(double)(c-x)/(double)(c-b);
	}
	if ((x == a) && (x == b))
		degree = 1;
	if ((x == a) && (x == c))
		degree = 1;
	return degree;
}

double F_tramf(uint16_t x, int p[]){
	int a=p[0], b=p[1], c=p[2], d=p[3];
	double degree=0;
	if ((x > a) && (x < d)) {
		if (x < b)
			degree=(double)(x-a)/(double)(b-a);
		else if (x>c)
			degree=(double)(d-x)/(double)(d-c);
		else
			degree=1;
	}
	if ((x == a) && (x == b))
		degree = 1;
	if ((x == c) && (x == d))
		degree = 1;
	return degree;
}

int compute_membership_degree(uint16_t a_temp, uint16_t soil_h){
	//coefficients for Temperature Input MF
	int coef_Small_Temp[]={0,0,desiredTemp};
	int coef_Medium_Temp[]={desiredTemp-1100,desiredTemp,desiredTemp+1100};
	int coef_High_Temp[]={desiredTemp,desiredTemp+1100,10000,10000};

	//coefficients for Humidity Input MF
	int coef_Dry_Hum[]={0,0,desiredHum};
	int coef_Fine_Hum[]={desiredHum-2000,desiredHum,desiredHum+2000};
	int coef_Wet_Hum[]={desiredHum,desiredHum+2000,10000,10000};

	double mdTemp[3];
	double mdHum[3];

	//membership degree to each set
	mdTemp[0]=F_trimf(a_temp,coef_Small_Temp);
	mdTemp[1]=F_trimf(a_temp,coef_Medium_Temp);
```

```
	mdTemp[2]=F_tramf(a_temp,coef_High_Temp);
	mdHum[0]=F_trimf(soil_h,coef_Dry_Hum);
	mdHum[1]=F_trimf(soil_h,coef_Fine_Hum);
	mdHum[2]=F_tramf(soil_h,coef_Wet_Hum);

	//compute activation degree of each rule
	int k=0;
	double adRule[no_Rules];
	for (int i=0;i<no_In_Temp;i++){
		for (int j=0;j<no_In_Hum;j++){
		adRule[k]=mdTemp[i]*mdHum[j];//AND
		k++;
		}
	}

	//defuzzification, mean average
	double denum=0, num=0;
	for (int i=0;i<no_Rules;i++){
		num = num+((double)pwm[rules[i]]*adRule[i]);
		denum = denum+adRule[i];
	}
	return(int)(num/denum);
}
```

main.c

```
/*
 * ATmega.c
 *
 * Created: 09/03/2024 11:54:58
 * Author : VladV
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdint.h>
#include <stdbool.h>

#define F_CPU 8000000UL
#include <util/delay.h>

#include "LCD.h"
#include "Fuzzy.h"

#define temp_hum_addr 0b1000100
#define ilum_addr 0b0100011

//#define BAUD_SOIL 4800
//#define BAUD_Wi_Fi 9600
unsigned int ubrrSoil = F_CPU/16/4800-1;
unsigned int ubrrWifi = F_CPU/16/9600-1;

#define DVI_LIGHT PIND6
#define HEAT_EN PINB5

unsigned char hello[] = "Hello :)";
unsigned char airTemp[] = "Air temp:  ";
unsigned char airHum[] =  "Air hum:   ";
unsigned char soilTemp[]= "Soil temp: ";
unsigned char soilHum[]=  "Soil hum:  ";
unsigned char soilPH[]=   "Soil PH:    ";
unsigned char soilNitro[]="Soil Nitro: ";
unsigned char soilPota[]= "Soil Potas: ";
unsigned char soilPhos[]= "Soil Phosp: ";
```

```
unsigned char airCO[]=    "CO con:  ";
unsigned char airNH3[]=   "NH3 con: ";
unsigned char airNO2[]=   "NO2 con: ";
unsigned char lightI[]=   "Light inten:";
unsigned char pwmS[]=   "PWM: ";
unsigned char err[]=    "Error";

uint8_t soil_meas[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x07, 0x04, 0x08};
uint8_t soil_res[20];

void initPorts();
void initUSART0(unsigned int);
void initUSART1(unsigned int);
void initI2C();
void start_I2C(unsigned char, unsigned char,uint8_t);
uint8_t read_w_ACK();
uint8_t read_w_NACK();
void stop_I2C();
void USART0_Tr(unsigned char);
unsigned char USART0_Rec();
void Flush_Rec0();
void USART1_Tr(unsigned char);
unsigned char USART1_Rec();
void Flush_Rec1();
void det_temp_hum();
void det_iulm();
void det_soil();
void initTimer0();
void initTimer2();
void initADC();
void transmWiFi(uint16_t);
void transmWiFi_1dec(uint16_t);
void displayLCD1Dec(uint16_t);
void displayLCD2Dec(uint16_t);
void computePWM();
void Error();

uint16_t temp, hum, ilum, soil_temp, soil_hum, soil_ph, soil_nitro, soil_phos, soil_pota,
nh3, no2, co;
int temperature, humidity, pwm_res;
bool water_level=1, count=0, automat=1, water_pump=0, gas_ok=0;
uint8_t duty_cycle, count_heat=0;
uint8_t no_of_samples=0, timer2_countisr=0;
uint16_t ADC0_CO[30], CO[30], ADC0_NH3[30], NH3[30], ADC0_NO2[30], NO2[30];

int main(void)
{
       //initializations
    initPorts();
       initI2C();
       initUSART0(ubrrSoil);
       initUSART1(ubrrWifi);
       initTimer0();
       initTimer2();
       initADC();
       LCDcommand(0x38);//init LCD
       LCDcommand(0x01);//clear display
       LCDcommand(0x0E);//place cursor
       LCDcommand(0x0C);//turn off cursor
       LCDdisplay(hello);
       LCDcommand(0x80);//move to the beginning of the 1st line

       sei();
       _delay_ms(1000);
```

```
    while (1)
    {
            if (count_heat==0)
                PORTB = PORTB & 0b11011111;//turn on heating

            det_temp_hum();

            det_soil();

            transmWiFi_bool(water_pump);

            det_iulm();

            count_heat++;
            if (count_heat>=1){

                PORTB = PORTB | (1<<HEAT_EN);//turn off HEAT_EN
                if (gas_ok==0)
                    TCCR2 = TCCR2 | (1<<WGM21) | (1<<CS22) | (1<<CS20);//turn on
timer 2, CTC mode, prescaler 1024
                if (gas_ok==1 && count_heat>=5){
                    count_heat=0;
                    gas_ok=0;
                }
            }

            LCDcommand(0x01);
            LCDdisplay(airCO);
            displayLCD0Dec(co);
            transmWiFi(co);
            LCDcommand(0xC0);//move to 2nd line
            LCDdisplay(airNH3);
            displayLCD0Dec(nh3);
            transmWiFi(nh3);
            LCDcommand(0x80);//force cursor to begin on first line
            _delay_ms(5000);

            LCDcommand(0x01);
            LCDdisplay(airNO2);
            displayLCD2Dec(no2);
            transmWiFi(no2);
            LCDcommand(0x80);//force cursor to begin on first line
            _delay_ms(5000);
        }
}

void initPorts(){

        DDRA = (1<<DDA7) | (1<<DDA6) | (1<<DDA5);//make EN output, RW output, RS output
        DDRD = (1<<DDD7) | (1<<DDD6) | (1<<DDD5);//make RE+DE, LIGHT_SENSOR, LIGHT_CTR
output
        PORTD = PORTD & 0b10011111;//make LIGHT_SENSOR, LIGHT_CTR low
        DDRE = (1<<DDE5);//make Wi_Fi_T_CTR output
        DDRE = DDRE & 0b10101111;//make Wi_Fi_R_INTR, WATER_LEVEL input
        PORTE = PORTE | (1<<PINE6) | (1<<PINE4);//activate pull-up
        PORTE = PORTE & 0b11011111;//make Wi_Fi CTR low
        DDRF = DDRF & 0b11110000;//make ADC inputs, high-Z
        DDRB = (1<<DDB5) | (1<<DDB4);//make HEAT_EN, WATER_CTR output
        PORTB = PORTB | (1<<HEAT_EN);//turn off HEAT_EN
        PORTB = PORTB & 0b11101111;//make WATER_CTR low
        //interr active on falling edge for WATER_LEVEL
        EICRB = 0x20;//ISC61 High, ISC60 Low
        EIMSK = 0x40;//activate INT6
}
```

```
void initUSART0(unsigned int BAUD_SOIL){

	//set baud rate
	UBRR0H = (unsigned char)(BAUD_SOIL>>8);
	UBRR0L = (unsigned char)(BAUD_SOIL);

	UCSR0B = (1<<RXEN0) | (1<<TXEN0);//enable RX0, TX0
	UCSR0C = (3<<UCSZ00);//8 data bits, 1 stop bit
}

void initUSART1(unsigned int BAUD_Wi_Fi){

	//set baud rate
	UBRR1H = (unsigned char)(BAUD_Wi_Fi>>8);
	UBRR1L = (unsigned char)(BAUD_Wi_Fi);

	UCSR1B = (1<<RXCIE1) | (1<<RXEN1) | (1<<TXEN1);//enable RX1 inter, RX1, TX1
	UCSR1C = (3<<UCSZ10);//8 data bits, 1 stop bit
}

void initI2C(){

	TWSR = 0;//TW Status Register, no Prescaler
	TWBR = (F_CPU - 16*100000) / (2 * 100000);//TW bit rate (to set SCL to 100kHz)
}

void initTimer0(){
	TCCR0 = 0;
	TIMSK = TIMSK | (1<<OCIE0);//enable timer interr
	OCR0 = 255;
}

void initTimer2(){
	TCCR2 = 0;
	TCCR2 = TCCR2 | (1<<WGM21);//CTC mode
	TIMSK = TIMSK | (1<<OCIE2);//enable timer interrupt
	OCR2 = 255;
}

void initADC(){
	ADMUX = ADMUX | (1<<REFS0);//AVCC reference voltage
	ADCSRA = ADCSRA | (1<<ADEN) | (1<<ADATE) | (1<<ADPS2) | (1<<ADPS0);//ADC enable,
Auto-trigger, prescaler = 32
	ADCSRB = 0;//free run mode
}

void det_temp_hum(){
	uint8_t status=0;

	start_I2C(temp_hum_addr,0,0x18);//measurement request; 0x18 = SLA+W has been
transmitted, ACK received
	stop_I2C();
	_delay_ms(100);
	start_I2C(temp_hum_addr,1,0x40);//data fetch; 0x40 - SLA+R transmitted, ACK
received

	hum = read_w_ACK();
	status = hum>>6;//first 2 MSB for status check
	if (status != 0x00)//check if rec data is ok
		Error();
	hum = hum & 0x3F;//delete first 2 MSB
	hum = (hum<<8) | read_w_ACK();
	temp = read_w_ACK();
```

```
	temp = (temp<<8) | (read_w_NACK() & 0xFC);//mask last 2 bits from 4th byte
	temp = temp>>2;//take off last 2 bits from 4th byte
	stop_I2C();

	temperature = ((temp / (pow(2,14)-1))*165 - 40)*100 - 400;
	humidity = (hum / (pow(2,14)-1))*10000;
	_delay_ms(5000);
	LCDcommand(0x01);
	LCDdisplay(airTemp);
	displayLCD2Dec(temperature);
	transmWiFi(temperature);
	LCDcommand(0xC0);//move to 2nd line
	LCDdisplay(airHum);
	displayLCD2Dec(humidity);
	transmWiFi(humidity);
	LCDcommand(0x80);//force cursor to begin on first line

}

void det_iulm(){

	PORTD = PORTD | (1<<PIND6);//put DVI "1"
	_delay_ms(10);

	start_I2C(ilum_addr,0,0x18);//measurement request
	TWDR = 0x10;//init auto-resolution mode

	TWCR = (1<<TWINT) | (1<<TWEN);//clear TWINT bit to start transmission
	while(!(TWCR & (1<<TWINT)));//wait for TWINT flag to set

	if((TWSR & 0xF8)!=0x18)//data byte transmitted and ACK has been received
		Error();

	stop_I2C();
	_delay_ms(120);

	start_I2C(ilum_addr,1,0x40);//data fetch
	ilum = read_w_ACK();
	ilum = (ilum<<8) | read_w_NACK();
	stop_I2C();

	ilum = ilum / 1.2;
	PORTD = PORTD & 0b10111111;//make LIGHT_SENSOR low
	_delay_ms(5000);
	LCDcommand(0x01);
	LCDdisplay(lightI);
	displayLCD0Dec4(ilum);
	LCDcommand(0x80);
	_delay_ms(5000);
	transmWiFi(ilum);
}

void det_soil(){
	DDRD = DDRD | (1<<DDD7);//make RE+DE output
	PORTD = PORTD | (1<<PIND7);//make RE+DE high -> transmit data from uC to sensor
	for (uint8_t i=0; i < 8; i++)
		USART0_Tr(soil_meas[i]);

	_delay_ms(10);
	DDRD = DDRD & 0b01111111;//make RE+DE high-Z-> receive data from sensor
	PORTD = PORTD & 0b01111111;//disable pull-up

	uint8_t addr = USART0_Rec();
	while (addr!=0x01)
```

```
		addr = USART0_Rec();
	uint8_t fctcode = USART0_Rec();
	uint8_t noB = USART0_Rec();
	if (noB < 14)
		Error();
	for (uint8_t i=0; i < noB+2; i++)
		soil_res[i] = USART0_Rec();

	if ((soil_res[noB]!=0x29) || (soil_res[noB+1]!=0x70))
		Error();

	soil_hum = (soil_res[0] << 8) | soil_res[1];
	soil_temp = (soil_res[2] << 8) | soil_res[3];
	soil_ph = (soil_res[6] << 8) | soil_res[7];
	soil_nitro = (soil_res[8] << 8) | soil_res[9];
	soil_phos = (soil_res[10] << 8) | soil_res[11];
	soil_pota = (soil_res[12] << 8) | soil_res[13];

	_delay_ms(5000);
	LCDcommand(0x01);
	LCDdisplay(soilHum);
	displayLCD1Dec(soil_hum);
	transmWiFi_1dec(soil_hum);
	LCDcommand(0xC0);//next line
	LCDdisplay(soilTemp);
	displayLCD1Dec(soil_temp);
	transmWiFi_1dec(soil_temp);
	LCDcommand(0x80);

	computePWM();

	_delay_ms(5000);
	LCDcommand(0x01);
	LCDdisplay(soilPH);
	displayLCD1Dec(soil_ph);
	transmWiFi_1dec(soil_ph);
	LCDcommand(0xC0);//next line
	LCDdisplay(soilNitro);
	displayLCD0Dec(soil_nitro);
	transmWiFi(soil_nitro);
	LCDcommand(0x80);

	_delay_ms(5000);
	LCDcommand(0x01);
	LCDdisplay(soilPhos);
	displayLCD0Dec(soil_phos);
	transmWiFi(soil_phos);
	LCDcommand(0xC0);//next line
	LCDdisplay(soilPota);
	displayLCD0Dec(soil_pota);
	transmWiFi(soil_pota);
	LCDcommand(0x80);

}

void computePWM(){
	if (automat==1 && water_level==1){
		pwm_res=compute_membership_degree(temperature,soil_hum*10);

		if (pwm_res > 20){//min duty cycle to power on
			duty_cycle = pwm_res*255/100;
			TCCR0 = TCCR0 | (1<<WGM01) | (1<<COM00) | (1<<CS02) |
(1<<CS01);//turn on timer 0, prescaler = 256, f=120Hz, toggle on compare match
			water_pump=1;
```

```
                        } else {
                        TCCR0 = 0;//turn off timer 0
                        water_pump=0;
                        PORTB = PORTB & 0b11101111;//make WATER_CTR low
                }
        }
}

void start_I2C(unsigned char slave_addr, unsigned char mode, uint8_t status){

        TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);//start condition. set the flags in TW
control register
        while (!(TWCR & (1<<TWINT)));//wait for TW Interr flag to set

        if ((TWSR & 0xF8) != 0x08)//check if start status ok (mask prescaler bits); 0x08 =
START condition transmitted
                Error();

        TWDR = (slave_addr<<1) | mode;//load slave addr. write op

        TWCR = (1<<TWINT) | (1<<TWEN);//clear TWINT bit to start transmission
        while(!(TWCR & (1<<TWINT)));//wait for TWINT flag to set

        if((TWSR & 0xF8)!=status)//check if ACK
                Error();
}

uint8_t read_w_ACK(){

        TWCR = (1<<TWINT) | (1<<TWEA) | (1<<TWEN);//set TWCR, send ACK at byte rec
        while(!(TWCR & (1<<TWINT)));
        return TWDR;
}

uint8_t read_w_NACK(){

        TWCR = (1<<TWINT) | (1<<TWEN);//set TWCR
        while(!(TWCR & (1<<TWINT)));
        return TWDR;
}

void stop_I2C(){

        TWCR = (1<<TWSTO) | (1<<TWINT) | (1<<TWEN);//stop condition
        while(TWCR & (1<<TWSTO));//wait until STOP cond exec
}

void Error(){
        LCDcommand(0x01);
        LCDdisplay(err);
        LCDcommand(0x80);
        _delay_ms(5000);
}

void USART0_Tr(uint8_t data){
        while(!(UCSR0A & (1<<UDRE0)));//wait for empty tr buffer (UDRE0 = Data Reg Empty
Flag)
        UDR0 = data;//put data in register
}

uint8_t USART0_Rec(){
        while(!(UCSR0A & (1<<RXC0)));//wait for data to be rec
        return UDR0;
}
```

```
void Flush_Rec0(){
	unsigned char d;
	while(UCSR0A & (1<<RXC0))
		d = UDR0;
}

void USART1_Tr(unsigned char data){
	while(!(UCSR1A & (1<<UDRE1)));//wait for empty tr buffer (UDRE1 = Data Reg Empty
Flag)
	UDR1 = data;//put data in register
}

unsigned char USART1_Rec(){
	while(!(UCSR1A & (1<<RXC1)));//wait for data to be rec
	return UDR1;
}

void Flush_Rec1(){
	unsigned char d;
	while(UCSR1A & (1<<RXC1))
		d = UDR1;
}

void transmWiFi(uint16_t x){
	PORTE = PORTE | (1<<PINE5);//make Wi_Fi CTR high
	USART1_Tr(x/100);
	USART1_Tr(x%100);
	PORTE = PORTE & 0b11011111;//make Wi_Fi CTR low
	while (PINE4==0);//wait for acknowledge of rec
}

void transmWiFi_1dec(uint16_t x){
	PORTE = PORTE | (1<<PINE5);//make Wi_Fi CTR high
	USART1_Tr(x/10);
	USART1_Tr(x%10);
	PORTE = PORTE & 0b11011111;//make Wi_Fi CTR low
	while (PINE4==0);//wait for acknowledge of rec
}

void transmWiFi_bool(bool x){
	PORTE = PORTE | (1<<PINE5);//make Wi_Fi CTR high
	USART1_Tr(x);
	PORTE = PORTE & 0b11011111;//make Wi_Fi CTR low
	while (PINE4==0);//wait for acknowledge of rec
}

ISR(INT6_vect){
	water_level = !water_level;//water level has changed
	EIMSK = EIMSK & 10111111;//mask interrupt

	if (water_level == 0){
		EICRB = EICRB | (1<<ISC61) | (1<<ISC60);//next time, on rising edge
		TCCR0 = 0;//stop WATER_CTR (stop timer)
		PORTB = PORTB & 0b11101111;//pull the pin LOW
		water_pump=0;
	}
	else {
		EICRB = EICRB | (1<<ISC61);
		EICRB = EICRB & 0b11101111;//next time, on falling edge
	}

	EIMSK = EIMSK | (1<<INT6);//enable interrupt 6 back
}
```

```
ISR(USART1_RX_vect){
	uint8_t rec = UDR1;
	if (rec==0)//turn off light
		PORTD = PORTD & 0b11011111;
		else if (rec==1)//turn on light
			PORTD = PORTD | (1<<PIND5);
		else if (rec==2)//turn on automatic control
			automat=1;
		else if (rec==3)//turn off automatic control
			automat=0;
		else if (rec==4 && water_level==1){//turn on water pump
			TCCR0 = TCCR0 | (1<<WGM01) | (1<<COM00) | (1<<CS02) |
(1<<CS01);//turn on timer 0, prescaler = 256
			duty_cycle = 127;//duty cycle of 50%
			water_pump=1;
		}
		else if (rec==5){//turn off water pump
			TCCR0 = 0;//stop WATER_CTR (stop timer)
			PORTB = PORTB & 0b11101111;
			water_pump=0;
		}
		else if (rec==6)
			desiredTemp = 1200;
		else if (rec==7)
			desiredTemp = 2500;
		else if (rec==8)
			desiredTemp = 3500;
		else if (rec==9)
			desiredTemp = 4800;
		else if (rec==10)
			desiredTemp = 7700;
		else if (rec==11)
			desiredHum = 1800;
		else if (rec==12)
			desiredHum = 4200;
		else if (rec==13)
			desiredHum = 6200;
		else if (rec==14)
			desiredHum = 8000;
	Flush_Rec1();
}

ISR(TIMER0_COMP_vect){
	if (count == 1){
		OCR0 = 255-duty_cycle;
		count=0;
	} else {
		OCR0 = duty_cycle;
		count=1;
	}
}


ISR(TIMER2_COMP_vect){
	uint16_t data;
	if (timer2_countisr==0){
	ADMUX = ADMUX & 0b11100000;
	ADMUX = ADMUX | (1<<MUX0);//read ADC1
	ADCSRA = ADCSRA | (1<<ADSC);//start ADC

	while ((ADCSRA & 0x10) == 0);//check ADIF bit (ADC conversion complete)

	data = ADCW;
```

```
CO[no_of_samples] = data;

ADCSRA = ADCSRA | (1<<ADIF);//clear ADIF bit
ADCSRA = ADCSRA & 0b10111111;//stop ADC

_delay_ms(1);

ADMUX = ADMUX & 0b11100000;
ADMUX = ADMUX | (1<<MUX4);//make difference between ADC0 and ADC1
ADCSRA = ADCSRA | (1<<ADSC);//start ADC

while ((ADCSRA & 0x10) == 0);//check ADIF bit (ADC conversion complete)

data = ADCW;

ADC0_CO[no_of_samples] = data;

ADCSRA = ADCSRA | (1<<ADIF);//clear ADIF bit
ADCSRA = ADCSRA & 0b10111111;//stop ADC

no_of_samples++;

if (no_of_samples>=20){
       timer2_countisr++;
       detCO();
       no_of_samples=0;
}
}

else if (timer2_countisr==1){

ADMUX = ADMUX & 0b11100000;
ADMUX = ADMUX | (1<<MUX1);//read ADC2
ADCSRA = ADCSRA | (1<<ADSC);//start ADC

while ((ADCSRA & 0x10) == 0);//check ADIF bit (ADC conversion complete)

data = ADCW;

NH3[no_of_samples] = data;
ADCSRA = ADCSRA | (1<<ADIF);//clear ADIF bit
ADCSRA = ADCSRA & 0b10111111;//stop ADC

_delay_ms(1);

ADMUX = ADMUX & 0b11100000;
ADMUX = ADMUX | (1<<MUX4) | (1<<MUX3);//read ADC0 difference with ADC2
ADCSRA = ADCSRA | (1<<ADSC);//start ADC

while ((ADCSRA & 0x10) == 0);//check ADIF bit (ADC conversion complete)

data = ADCW;

ADC0_NH3[no_of_samples] = data;

ADCSRA = ADCSRA | (1<<ADIF);//clear ADIF bit
ADCSRA = ADCSRA & 0b10111111;//stop ADC

no_of_samples++;

if (no_of_samples>=20){
       detNH3();
       timer2_countisr++;
```

```
			no_of_samples=0;
			//transmWiFi(nh3);
		}
		}
		else if (timer2_countisr==2){
		ADMUX = ADMUX & 0b11100000;
		ADMUX = ADMUX | (1<<MUX1) | (1<<MUX0);//read ADC3
		ADCSRA = ADCSRA | (1<<ADSC);//start ADC

		while ((ADCSRA & 0x10) == 0);//check ADIF bit (ADC conversion complete)

		data = ADCW;

		NO2[no_of_samples] = data;

		ADCSRA = ADCSRA | (1<<ADIF);//clear ADIF bit
		ADCSRA = ADCSRA & 0b10111111;//stop ADC

		_delay_ms(1);

		ADMUX = ADMUX & 0b11110000;//read ADC0
		ADCSRA = ADCSRA | (1<<ADSC);//start ADC
		while ((ADCSRA & 0x10) == 0);//check ADIF bit (ADC conversion complete)

		data = ADCW;

		ADC0_NO2[no_of_samples] = data - NO2[no_of_samples];

		ADCSRA = ADCSRA | (1<<ADIF);//clear ADIF bit
		ADCSRA = ADCSRA & 0b10111111;//stop ADC
		no_of_samples++;

		if (no_of_samples>=20){
			detNO2();
			timer2_countisr=0;
			no_of_samples=0;
			gas_ok=1;
			TCCR2 = 0;//stop timer
		}
		}
}

uint16_t average(uint16_t x[], uint16_t y[], int multipl){
		uint16_t sum[2]={0};
		for (uint8_t i=0; i<no_of_samples;i++){
			sum[0] = sum[0] + x[i];
			sum[1] = sum[1] + y[i];
		}

		for (uint8_t i=0;i<2;i++)
			sum[i] = sum[i]/no_of_samples;
		uint16_t res1 = sum[0]*3.3/pow(2,10)*10000;
		uint16_t res2 = sum[1]*3.3/pow(2,10)*10000;

		uint16_t Current = (double)res1/56;

		uint16_t R = (double)res2/Current*multipl;

		return R;
}

void detCO(){
		uint16_t R0 = 2;//sesnsing resistance in clean air
		uint16_t Rs = average(ADC0_CO,CO,1);
```

```
        double rap = (double)Rs/(double)R0;
        if (rap <=0.07)
                co = -15000*rap + 1150;
        else if (rap<=0.5)
                co = -209*rap + 114;
        else if (rap <=4)
                co = -2.5*rap + 11;
}

void detNH3(){
        uint16_t R0 = 862;//sesnsing resistance in clean air
        uint16_t Rs = average(ADC0_NH3,NH3,1);
        double rap = (double)Rs/(double)R0;
        nh3 = Rs;/*
        if (rap <=0.25)
                nh3 = -500*rap + 135;
        else if (rap<=0.8)
                nh3 = -16*rap + 14;*/
}

void detNO2(){
        uint16_t R0 = 5025;//sesnsing resistance in clean air
        uint16_t Rs = average(ADC0_NO2,NO2,100);
        double rap = (double)Rs/(double)R0;
        if (rap <=0.15)
                no2 = (0.9*rap - 0.035)*100;
        else if (rap<=6)
                no2 = (0.153*rap + 0.07)*100;
        else if (rap<=40)
                no2 = (0.15*rap + 0.09)*100;
}
```

9.9 ESP-12E code

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>           // WiFi control
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <math.h>

#define CTR 14
#define ACK_RX 12

// declaration of wi-fi credentials
const char* wifiSSID = "Vlad";
const char* wifiPassword = "Vld1234VV";

// ThingsBoard credentials
String tbHost = "demo.thingsboard.io";
String tbToken = "0mxqRJLizu154m1ZfAwY";

WiFiClient client;
PubSubClient mqtt(client);

SoftwareSerial SerialPort(4,5);//comm with ATmega64
```

```
int transmWiFifreq=0;
bool waterPumpState, lightState=0;
double desiredTemp=22;
double desiredHum=40;
bool tempAlarm=false;
bool Auto=true;

// functions declarations
void connectWifi();
void tbReconnect();
void on_message(const char *topic, byte *payload, unsigned int length);

String getInfoInt(String n, int x){
  JsonDocument data;
  data[n] = x;
  char payload[256];
  serializeJson(data,payload);//prepare for sending (serialize data into payload)
  String strPayload = String(payload);
  Serial.print("Get info: ");
  Serial.println(strPayload);
  return payload;
}

String getInfoBool(bool x){
  JsonDocument data;
  data = x;
  char payload[256];
  serializeJson(data,payload);//prepare for sending
  String strPayload = String(payload);
  Serial.print("Get info: ");
  Serial.println(strPayload);
  return payload;
}

double Prec2(double x){
  int y = (int)(x*100)/100;
  int z = (int)(x*100)%100;
  return (double)(y)+(double)(z)/100;
}

String getInfoDoublewString(String n, double x){
  JsonDocument data;
  data[n] = Prec2(x);
  char payload[256];
  serializeJson(data,payload);//prepare for sending
  String strPayload = String(payload);
  Serial.print("Get info: ");
  Serial.println(strPayload);
  return payload;
}
```

```
String getInfoDoublewoString(double x){
  JsonDocument data;
  data = Prec2(x);
  char payload[256];
  serializeJson(data,payload);//prepare for sending
  String strPayload = String(payload);
  Serial.print("Get info: ");
  Serial.println(strPayload);
  return payload;
}

double readData2Dec(){
  int dInt, dSubu;
  while (digitalRead(CTR)!=HIGH){
      yield();  //loop lasts longer than usual and not activate watchdog
  }

  dInt = SerialPort.read();

  dSubu = SerialPort.read();

  //send acknowledge of rec
  digitalWrite(ACK_RX,HIGH);
  delay(2);
  digitalWrite(ACK_RX,LOW);

  return (double)(dInt) + (double)(dSubu%100)/100;
}

double readData1Dec(){
  int dInt, dSubu;
  while (digitalRead(CTR)!=HIGH){//wait for ATmega to transmit data
      yield();
  }

  dInt = SerialPort.read();

  dSubu = SerialPort.read();

  //send acknowledge of rec
  digitalWrite(ACK_RX,HIGH);
  delay(2);
  digitalWrite(ACK_RX,LOW);

  return (double)(dInt) + (double)(dSubu%10)/10;
}

double readData0Dec(){
  int dInt, dSubu;
```

```
  while (digitalRead(CTR)!=HIGH){//wait for ATmega to transmit data
      yield();
  }

  //read data
  dInt = SerialPort.read();

  dSubu = SerialPort.read();

  //send acknowledge of rec
  digitalWrite(ACK_RX,HIGH);
  delay(2);
  digitalWrite(ACK_RX,LOW);

  return (double)(dInt)*100 + (double)(dSubu);
}

void readPumpState(){
  bool x;
  while (digitalRead(CTR)!=HIGH){
      yield();
  }

  x = SerialPort.read();

  //send acknowledge of rec
  digitalWrite(ACK_RX,HIGH);
  delay(2);
  digitalWrite(ACK_RX,LOW);

  if (x!=waterPumpState){
    waterPumpState=x;
    mqtt.publish("v1/devices/me/attributes",getInfoInt("Water Pump
On/Off",waterPumpState).c_str());
  }
}

void setup() {
  Serial.begin(9600);
  SerialPort.begin(9600); // Set the baud rate and pins for SoftwareSerial
  pinMode(CTR,INPUT);
  pinMode(ACK_RX,OUTPUT);
  digitalWrite(ACK_RX,LOW);
  connectWifi();
  mqtt.setServer(tbHost.c_str(), 1883); // set the server for MQTT communication using
specific port 1883
  mqtt.setCallback(on_message); // set the callback method for receiving data from the
platform
  delay(500);
}
```

```
void loop() {
double data;

  if (!mqtt.connected()){//connection to platform
    tbReconnect();
  }

  data = readData2Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Air
Temperature",data).c_str());
    Serial.println(data);
    Serial.println("Sending data to Thingsboard");
  }

  data = readData2Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Air
Humidity",data).c_str());
    Serial.println(data);
    Serial.println("Sending data to Thingsboard");
  }

  mqtt.loop();

  data = readData1Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Soil
Humidity",data).c_str());
    Serial.println(data);
  }

  data = readData1Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Soil
Temperature",data).c_str());
    Serial.println(data);
  }

  mqtt.loop();

  data = readData1Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Soil PH",data).c_str());
    Serial.println(data);
  }

  data = readData0Dec();
  if (transmWiFifreq==0){
```

```
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Soil
Nitrogen",data).c_str());
    Serial.println(data);
  }

  mqtt.loop();

  data = readData0Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Soil
Phosphorus",data).c_str());
    Serial.println(data);
  }

  data = readData0Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Soil
Potassium",data).c_str());
    Serial.println(data);
  }

  mqtt.loop();

  readPumpState();

  /*

  if ((temp < tempThreshold - 0.2) && (tempAlarm!=true)){//set alert for temperature
    mqtt.publish("v1/devices/me/attributes",getInfoInt("TempAlert",1).c_str());
    tempAlarm = true;
  }
   else if (temp > tempThreshold + 0.4){
    mqtt.publish("v1/devices/me/attributes",getInfoInt("TempAlert",0).c_str());
    tempAlarm = false;
   }*/

  data = readData0Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Light
Intensity",data).c_str());
    Serial.println(data);
  }

  mqtt.loop();


  data = readData0Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Air CO",data).c_str());
    Serial.println(data);
```

```
  }

  data = readData0Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Air NH3",data).c_str());
    Serial.println(data);
  }

  mqtt.loop();

  data = readData2Dec();
  if (transmWiFifreq==0){
    mqtt.publish("v1/devices/me/telemetry",getInfoDoublewString("Air NO2",data).c_str());
    Serial.println(data);
  }

  transmWiFifreq++;  //count how often a Wi-Fi transmission is made
  if (transmWiFifreq>5)
    transmWiFifreq=0;

}

void tbReconnect(){
  while (!mqtt.connected()){
    if (WiFi.status() != WL_CONNECTED)
      connectWifi();

    Serial.println("connecting to thingsboard ... ");
    if (mqtt.connect(tbHost.c_str(), tbToken.c_str(),NULL)){
      Serial.println("Thingsboard Connected!");
      mqtt.subscribe("v1/devices/me/rpc/request/+");// subscription to RPC request (to get
data from the platform)
      mqtt.publish("v1/devices/me/attributes",getInfoInt("Water Pump
On/Off",waterPumpState).c_str());
      mqtt.publish("v1/devices/me/attributes",getInfoInt("Light
On/Off",lightState).c_str());

      }
    else{
      Serial.println("Thingsboard connection failed");
      Serial.println("Retrying in 5 seconds...");
      delay(5000);
    }
  }
}

void connectWifi() {
  Serial.println("Connecting To Wifi");
  WiFi.begin(wifiSSID, wifiPassword);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    Serial.print(".");
    delay(500);
  }

  Serial.println("Wifi Connected");
  /*Serial.println(WiFi.SSID());
  Serial.println(WiFi.RSSI());
  Serial.println(WiFi.macAddress());
  Serial.println(WiFi.localIP());
  Serial.println(WiFi.gatewayIP());
  Serial.println(WiFi.dnsIP());
  */

}

void on_message(const char *topic, byte *payload, unsigned int length) {
  Serial.println("On message");
  char json[length + 1];
  strncpy(json, (char *)payload, length);
  json[length] = '\0';

// display topic and message received
  Serial.print("Topic: ");
  Serial.println(topic);
  Serial.print("Message: ");
  Serial.println(json);

  JsonDocument data;
  deserializeJson(data,json);//deserialize the message

  if (data.isNull())
  {
    Serial.println("parseObject() failed");
    return;
  }

  // Check request method
  String methodName = String((const char*)data["method"]);

  if (methodName.equals("setPumpState")){
    if (Auto!=true){
      waterPumpState=data["params"];
      if (waterPumpState==true)//send water pump state to ATmega
        SerialPort.write(4);
      else
        SerialPort.write(5);
    }

// send response to platform by setting the pump state (topic + message)
    String responseTopic = String(topic);
```

```
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoBool(waterPumpState).c_str());
    mqtt.publish("v1/devices/me/attributes",getInfoInt("Water Pump
On/Off",waterPumpState).c_str());
  }
  else if (methodName.equals("setLightState")){
    lightState=data["params"];
    SerialPort.write(lightState);
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoBool(lightState).c_str());
    mqtt.publish("v1/devices/me/attributes",getInfoInt("Light
On/Off",lightState).c_str());
  }
  else if (methodName.equals("setDesiredTemp")){
    desiredTemp=data["params"];
    if (desiredTemp < 20)
      SerialPort.write(6);
    else if (desiredHum < 30)
      SerialPort.write(7);
    else if (desiredHum < 40)
      SerialPort.write(8);
    else if (desiredTemp < 55)
      SerialPort.write(9);
    else
      SerialPort.write(10);
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoDoublewoString(desiredTemp).c_str());
  }
  else if (methodName.equals("getDesiredTemp")){
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoDoublewoString(desiredTemp).c_str());
  }
  else if (methodName.equals("setDesiredHum")){
    desiredHum=data["params"];
    if (desiredHum < 30)
      SerialPort.write(11);
    else if (desiredHum < 55)
      SerialPort.write(12);
    else if (desiredHum < 70)
      SerialPort.write(13);
    else
      SerialPort.write(14);
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoDoublewoString(desiredHum).c_str());
  }
  else if (methodName.equals("getDesiredHum")){
```

```
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoDoublewoString(desiredHum).c_str());
  }
  else if (methodName.equals("getAuto")){
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoBool(Auto).c_str());
  }
  else if (methodName.equals("setAuto")){
    Auto = data["params"];
    if (Auto==true)//send automatic control state to ATmega
      SerialPort.write(2);
    else
      SerialPort.write(3);
    String responseTopic = String(topic);
    responseTopic.replace("request", "response");
    mqtt.publish(responseTopic.c_str(), getInfoBool(Auto).c_str());
  }
}
```

9.10 SSET Article

System for Automatic Greenhouse

Vlad VELICIU
Faculty of Electronics, Telecommunications and Information Technology
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
vladveliciu16@gmail.com

Conf. Dr. Ing. Liviu VIMAN
Applied Electronics Department
Faculty of Electronics, Telecommunications and Information Technology, Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Liviu.VIMAN@ael.utcluj.ro

***Abstract*—This paper proposes an implementation of a system for automatic greenhouses, exposing a solution for both monitoring and controlling major parameters. It consists of an ATmega64A microcontroller, five sensors, two control parts and a Wi-Fi module, all being embedded on a PCB. For safety reasons, galvanic isolation between the components that must be connected outside the PCB and the ones soldered on it, was designed to avoid hazards on the lines outside the board to get to sensitive components. Communication is achieved via the Wi-Fi module that uses the MQTT protocol to exchange information between ATmega64A and an IoT platform – Thingsboard.**

Keywords—greenhouse, control system, IoT platform, PCB

I. Introduction

Greenhouses represent a solution for the climate change we experience nowadays, as it has a huge impact on the agriculture and thus on the food chain around the world. The parameters that are of significance for a healthy and sustainable growth of the plants become more and more difficult to control in an environment that experiences extreme events. This fact enhances the nowadays need to develop solutions in order to overcome this challenge and to keep food security [1].

The proposed implementation consists of monitoring some parameters that are significant for a greenhouse: air temperature, air humidity, light intensity, NO2, NH3 and CO concentration, as well as soil parameters (temperature, moisture, PH, Nitrogen, Phosphorus, Potassium). The data gathered is interpreted by the microcontroller and then sent to the Wi-Fi module and finally to the IoT platform where the user can visualize it. The control of a water pump and also a relay that will power up a circuit for lights inside the greenhouse is available. As the water that will be fed to the soil by the water pump will be in a tank, a water level sensor is present to avoid the case in which the water pump draws water from the tank when it's empty. All the data will be displayed also locally, on an LCD. All of these elements are incorporated on a 4-layer PCB (Layer 1 and 4 - signal, Layer 2 - Ground, Layer 3 - Power). The signal traces are routed on layers 1 and 4 with a width of 0.3…0.4mm, the copper weight on these layers is 35µm and the distance to the adjacent layer that act as reference plane is 0.21mm, reducing thus the inductive crosstalk.

From the block diagram below (Fig. 1), the relations between the microcontroller and the sensors, Wi-Fi module and control elements can be seen. For the devices that need to be placed further away from the board, the soil quality sensor, water level sensor, water pump and light control circuit, a galvanic isolated area on the PCB was designed for protection reasons. On this part of the board all the components are powered by 5V, while on the "small signal" one by 3.3V.

Figure 1 Block Diagram

II. Circuit Design

A. The microcontroller ATmega 64A

The microcontroller is used for I2C communication, UART data transfer with the soil quality sensor and the Wi-Fi module, analog-digital conversion, for sending commands and receiving interrupts.

B. The air temperature, humidity and light intensity sensors

These sensors use the I2C interface to communicate. On the PCB, the I2C traces were guarded with ground vias in order to protect the adjacent lines from the EM emissions that can be caused by the relatively high frequency of 100kHz.

C. Soil Quality Sensor

This sensor uses RS485 communication, and it is connected outside the board. So, a galvanic isolation through optocouplers is designed, as shown in Fig. 2. The MAX485 chip is used to interface the serial communication of the microcontroller with the RS485; basically, a string of bytes is converted by the chip into the differential RS485 communication and vice-versa. The routing of these lines on the board was made according to the characteristic impedance of 120ohm. A fail-safe configuration was implemented, through a termination resistance on the chip side [2]. The differential traces were routed with a width of 0.12mm and a gap of 0.2mm, to achieve the characteristic impedance.

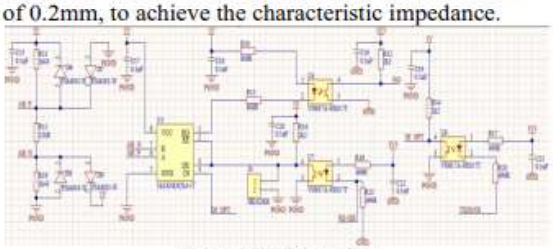

Figure 2 RS485 interface

D. Air quality sensor

The MICS-6814 sensor is an analog, metal oxide sensor. For each of the three gases, heating is needed before

performing the measurement via the ADC. The heating layer is seen as a resistance, a heating voltage being necessary for each gas. The sensitive layer changes its resistance with the gas concentration. After a certain heating time (30s), the sensitive resistance is measured. At first time use, an environment in which the gas concentrations are known is needed to correlate the measured resistance value with them, as only the sensitivity-gas concentration characteristic is known (Rs/Ro where Rs is the sensitive resistance and Ro the reference air resistance that is determined for further use).

E. Water pump and light control (relay)

These are commanded by the microcontroller via optocouplers. To get a higher current for the water pump and for the relay, and to avoid dissipating much power on the transistor from the output of the optocoupler, another npn transistor is used. The collector current of the transistor from the optocoupler is set to 5mA, this feeding the base of the "power" transistor sustaining the necessary current for the load. The transistor needs to switch with a certain frequency and, a Schottky diode is inserted between the collector of the transistor and the power line, as shown in Fig. 3, to avoid ringing at switching, the load having inductive elements. These diodes provide a very small recovery time (time needed to evacuate the charges when the diode gets off), so that they can rapidly drain the switching noise.

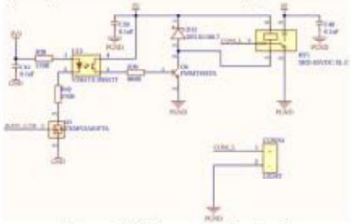

Figure 3 Relay control circuit

F. Optocoupling design

The resistances designed to control the current through the LED input are disposed symmetrical around its pins (Fig. 3 and Fig. 4) to improve the common-mode rejection [3].

G. Water Level sensor

This sensor provides at the output a voltage that corresponds to the sensing water level (maximum 2.5V). To transmit the change of water level in a tank (empty/full) through the optocoupler, a hysteresis comparator, shown in Fig. 4, was designed, with thresholds that were set, experimentally, at 2.2V and 2V. The switching between high and low voltage level on the hysteresis triggers an interrupt.

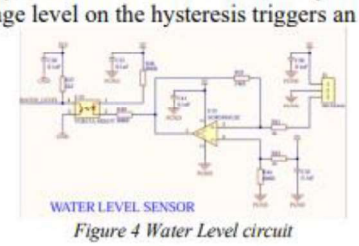

Figure 4 Water Level circuit

H. Wi-Fi module

The ESP-12E Wi-Fi module uses the 2.4GHz band and it is used to transmit and receive data from an IoT platform. This is performed via the MQTT protocol; entities are handled by a broker which in this case is represented by the IoT platform [4]. For data transfer with the microcontroller, the UART protocol is used, and two lines for handshaking. As the module consumes significant current during reception/transmission (up to 170mA), an electrolytic capacitor is needed to act as local energy reservoir. On the PCB a cut-out under the area of the antenna was performed to reduce the influence of the 2.4GHz field on the traces.

I. Power supplies

The 3.3V and 5V supplies are formed using 2 LDO (Low-Dropout) chips. The maximum current delievered by the 3.3V one is 750mA and for the 5V LDO, 400mA. Thermal vias were designed under the thermal pad of both chips to provide a second heat path that has a lower thermal resistance.

III. Results

Fig. 5 shows a simulation of the collector-emitter voltage on the transistor (purple) with a Schottky diode (left), and without (right - overshoot that can damage it).

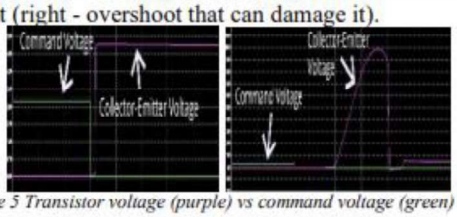

Figure 5 Transistor voltage (purple) vs command voltage (green)

Fig. 6 shows the I2C communication signals with HS3002 measured from the board when a measurement result is requested.

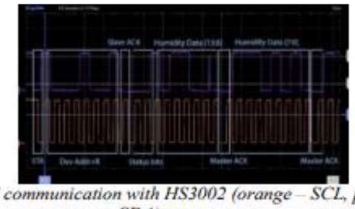

Figure 6 I2C communication with HS3002 (orange – SCL, purple - SDA)

Fig. 7 shows the 3D model of the board with all the components and elements described above.

Figure 7 3D model of the board

References

[1] Introductory Chapter: Climate Change and Climate-Smart Greenhouses. Written By Ahmed A. Abdelhafez,Mohamed H.H. Abbas, Shawky M. Metwally, Hassan H. Abbas, Amera Sh. Metwally, Khaled M. Ibrahim, Aya Sh. Metwally, Rasha R.M. Mansour and Xu Zhang.

[2] https://resources.altium.com/p/serial-communications-protocols-rs-485

[3] https://www.renesas.com/us/en/document/apn/an1991-isolating-rs-485-interfaces-high-speed-digital-optocouplers

[4] https://www.hivemq.com/mqtt


TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
EUROPEAN UNIVERSITY OF TECHNOLOGY
Faculty of Electronics, Telecommunications and Information Technology
THE 19th STUDENT SYMPOSIUM ON ELECTRONICS AND TELECOMMUNICATIONS
May 24, 2024
1st Prize
Vlad VELICIU
author of the paper
"System for Automatic Greenhouse"
coordinated by
Assoc.prof. Liviu VIMAN
Professor Ovidiu POP
Dean
Professor Dorin PETREUȘ
Chairman

10 Author's CV

europass

Vlad Veliciu

Home : Cluj-Napoca, Romania

Email: vladveliciu16@gmail.com **Phone:** (+40) 728301400

WhatsApp Messenger: +40728301400

Facebook: https://www.facebook.com/vlad.veliciu/

LinkedIn: https://www.linkedin.com/in/vlad-veliciu-a8587a210/

Git Hub: https://github.com/VladVeliciu

Instagram: https://www.instagram.com/vladveliciu/

Gender: Male **Date of birth:** 16/05/2001 **Nationality:** Romanian

EDUCATION AND TRAINING

[28/09/2020 – Current] **Bachelor's degree**

Technical University of Cluj-Napoca https://etti.utcluj.ro/acasa.html

Address: str. George Baritiu, nr. 26-28, Cluj-Napoca, Romania | **Field(s) of study:** Electronics, Telecommunications and Information Technology

Applied Electronics, English language

[12/09/2016 – 01/07/2020] **Baccalaureate diploma**

Stephan Ludwig Roth Theoretical Highschool https://www.liceulroth.ro/

Address: Piata George Enescu, no. 7, 551018, Medias, Romania | **Field(s) of study:** Mathematics-Informatics, German language

Mathematics, Informatics, Physics, German, English

LANGUAGE SKILLS

Mother tongue(s): Romanian

Other language(s):

German

LISTENING C1 **READING** C1 **WRITING** C1

SPOKEN PRODUCTION C1 **SPOKEN INTERACTION** C1

English

LISTENING B2 **READING** C1 **WRITING** B2

SPOKEN PRODUCTION C1 **SPOKEN INTERACTION** C1

Levels: A1 and A2: Basic user; B1 and B2: Independent user; C1 and C2: Proficient user

DIGITAL SKILLS

My Digital Skills

C, C++, C# | C/C++ for Microcontrollers | Altium (PCB Design) | Assembly (8051) | microprocessor systems | VHDL language (basic) | MATLAB | Microsoft Office | Lt-spice | Intellij Idea | Software Engineering | Visual Studio | HTML | Python | SQL/MYSQL | Java | Object-Oriented Programming | Github | CSS

LICENSES & CERTIFICATIONS

[31/05/2024 – Current] **Certificate of Competence in PCB design**

APTE (Association for Promoting Electronics Technology)

[26/02/2021 – Current] **Translator Certificate - German-Romanian, Romanian-German - Technics, Electronics-Telecommunications domains**

Institutul National pentru Cercetare si Formare Culturala

[10/06/2020 – Current] **Deutsches Sprachdiplom der Kultusministerkonferenz - Zweite Stufe (DSD II) - C1 Niveau**

Kultusministerium

[02/09/2019 – Current] **Cambridge English Level 1 Certificate in ESOL International**

Cambridge Assessment English

PROJECTS

[10/07/2023 – 10/09/2023] **Soil Quality Sensor**

- team project
- a system composed by an ESP32-S3 microcontroller that receives data from a 7-in-1 soil quality sensor (C language used for programming)
- data is then transmitted via Wi-Fi using using the MQTT protocol to a Raspberry Pi 4 that also acts as an access point.
- the IoT platform to which the data is transmitted is ThingsBoard, this being installed on the Raspberry Pi
- a PCB was developed for the ensamble formed by the microcontroller-sensor-transceiver
- a report was written and a comparison with other implementations of this project was made
- project developed at Universitat de Valencia, Spain

Link: https://github.com/VladVeliciu/Soil-Quality-Sensor

[05/12/2018 – 10/03/2020] **Code Warriors Robotics Team**

Founding member

Worked in the Mechanics department in the team

Our greatest achievements in the two seasons:

- 2018/2019, Timisoara Qualifying Tournament: Collins Aerospace Innovate Award 2nd place

- 2018/2019, Bucharest Championship: participation

- 2019/2020, Cluj-Napoca Qualifying Tournament: Finalist Alliance Award - Captain

- 2019/2020, Cluj-Napoca Qualifying Tournament: Inspire Award 3rd place

Link: https://www.facebook.com/wearecodewarriors

HONORS AND AWARDS

First Mention (4th place) at the International Student Contest "Interconnection Techniques in Electronics", TIE 2024

Design of Electronic Modules and Assemblies, 26th of April, 2024

Mention at the International Student Contest "Interconnection Techniques in Electronics", TIE 2023

Design of Electronic Modules and Assemblies, 20th of October, 2023

Participation at "Tudor Tanasescu" Annual National Contest

Analog Integrated Circuits Section, Bucharest, 1st of April, 2023

Roberto Rocca Tenaris Scholarship

Won one of the 40 scholarships for engineering students in Romania, 21st of November, 2022

CONFERENCES AND SEMINARS

[24/05/2024 – 24/05/2024] **SSET (Student Symposium on Electronics and Telecommunications), 19th edition**
Cluj-Napoca

Won the first prize as the author of the paper "System for Automatic Greenhouse" coordinated by Assoc. prof. Liviu Viman

Link: https://etti.utcluj.ro/files/Acasa/Site/SSET/Brosura SSET 2024 site.pdf

COURSES

[12/09/2022 – 16/09/2022] **Big Data Management and Analytics**

- processing structured data in Python (using Pandas) and using SQL query language (using MySQL)
- perform computations involving large databases and basic parallel data processing using Apache Spark
- storing, querying, processing semi-structured data in Python/in a NoSQL database (MongoDB)

VOLUNTEERING

[02/08/2021 – 30/09/2021] **Tourist Guide** St Margaret's Church Medias

[30/10/2020 – 20/02/2021] **Writer and editor at the German redaction** POV21

Link: https://de.pov21.com/author/vlad-veliciu/

[19/06/2019 – 15/09/2019] **Tourist Guide** St Margaret's Church Medias

[01/07/2018 – 15/09/2018] **Tourist Guide** St Margaret's Church Medias

[15/06/2017 – 01/09/2017] **Archiving Books at the Synagogue in Medias**
Synagogue Medias, Mihai Eminescu Trust Romania

HOBBIES AND INTERESTS

Hiking

Travelling

Conform cu originalul