Proiect 2 - Documentatie
Minisistem pentru irigat plante

Chisalescu Bogdan si Floricel Antonio - Stefan
Grupa 432A

Coordonator stiintific: S.l. dr. ing. Valentin Stoica

Cuprins

1 Cerinta de proiectare

Implementarea unui minisistem pentru irigat plante folosind microcontrolerul Atmel AVR ATmega 164A. Realizarea fizica trebuie efectuata pe o placa de cablaj imprimat cu componente SMD furnizata de catre facultate, insa datorita conditiilor epidemiologice si masurilor luate de Conducerile ETTI si UPB aceasta modalitate de realizare nu a fost posibila; masurile de distantare sociala si invatamant in regim "online" au facut impractic accesul la aparatura necesara, astfel am decis realizarea proiectului pe placi de prototipare si cu componente THT. Realizarea in formatul SMD + placa de circuit imprimat s-ar fi putut realiza de catre un singur membru al echipei insa in acest mod contributia asupra proiectului ar fi fost sever dezechilibrata. Intelegem ca in urma acestei decizi se pierd anumite avantaje precum stabilitatea cu temperatura, zgomotul redus, costul redus, etc.

2 Solutia de implementare propusa

Pentru a implementa minisistemul de irigat plante vom avea nevoie de doua clase principale de componente: senzori si actuatori. Actiunea de a iriga plantele consta in transferul unei cantitati de apa dintr-un rezervor catre plante. Pentru a decide momentul irigarii si a o efectua este nevoie de prelucrarea marimilor fizice implicate si anume umiditatea solului si disponibilitatea apei din rezervor reprezentata prin nivel (inaltimea coloanei de apa). Pentru a extrage si cuantifica aceste marimi vom folosi un senzor de umiditate a solului si unul de nivel. Pentru transportul apei de la rezervor la plante vom folosi ca actuator o minipompa de apa. Pentru a facilita interactiunea cu sistemul al unui posibil utilizator vom dezvolta o aplicatie desktop ce va asigura transmisia de date de configurare plus o serie de informatii despre starea sistemului in timp real.

3 Schema bloc a sistemului

Schema bloc a sistemului reprezinta arhitectura generala si descrie modul conceptual de interconectare a componentelor. Se poate observa o comunicatie unilaterala intre blocuri precum senzorii si microcontroler si o comunicatie bilaterala intre microcontroler si aplicatia utilizatorului, acestea semnificand sensul de transmitere al informatiei.

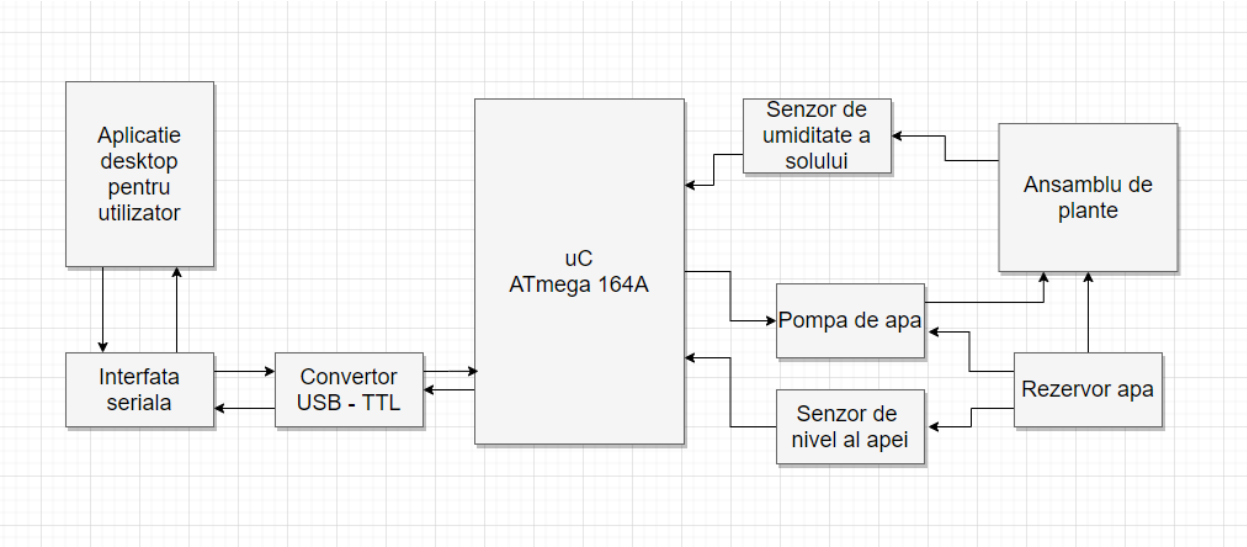

Figure 1: Schema bloc a minisistemului de irigat plante

4 Incarcarea codului pe microcontroler

Microcontrolerul dispune de mai multe modalitati prin care poate fi programat. Cele care sunt de interes sunt programarea folosind un programator dedicat si programarea folosind un bootloader. Utilizarea programatorului dedicat este simpla intrucat este necesara doar conexiunea sa catre PC prin portul USB si realizarea conxiunilor catre pinii dedicati ai microcontrolerului. Utilizarea bootloaderului presupune intai scrierea acestuia intr-o memorie nonvolatila, spatiul rezervat acestuia fiind in adresele superioare are memoriei Flash de program. Dupa incarcarea bootloaderului nu mai este

nevoie de programatorul dedicat deoarece in procesul de programare al microcontrolerului bootloaderul preia controlul folosind ce interfata seriala are la dispozitie, incarca aplicatia ce va rula pe microcontroler in memroia de program si ulterior ii cedeaza acesteia controlul executiei. In aceasta lucrare am optat pentru incarcarea codului cu un programator dedicat deoarece ne confera mai multe functionalitati. Am ales pentru elaborarea codului mediul de dezvoltare Atmel Studio 7 acesta fiind creat special pentru programarea microcontrolerelor fabricate de Atmel. Cu ajutorul Atmel Studio si compilatorului GCC vom crea un fisier cu extensia .hex ce va fi incarcat pe microcontroler de catre softul "avrdude" folosind comanda "avrdude -c usbasp -p m164p -U flash:w:$(ProjectDir)Debug $(TargetName).hex:i". Deoarece avrdude nu ofera suport pentru modelul ATmega 164A, am modificat in fisierul de configurare al avrdude semnatura dispozitivului alegand cel mai apropiat dispozitiv ca arhitectura de ATmega 164A, acesta fiind ATmega 164P.

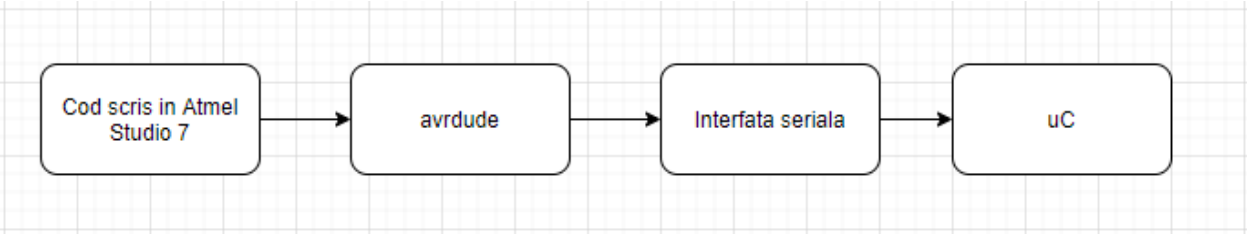

Figure 2: Modul in care se incarca codul pe uC

5 Cuplarea unui oscilator extern

Atmega 164A pune la dispozitie o multitudine de optiuni cand vine vorba de alegerea sursei pentru ceasul intern; acesta vine echipat cu un oscilator RC intern calibrat la frecventa de 8 MHz, iar pentru ceasul de sistem aceasta frecventa este impartita la 8, de unde rezulta 1 MHz, astfel in mod implicit microcontrolerul ruleaza la o frecventa de ceas de 1 MHz. Deoarece aceasta frecventa de 1 MHz este in anumite situatii un factor limitator de performanta vom cupla un oscilator extern cu cristal de quartz de 20 MHz. Pentru a determina microcontrolerul sa foloseasca acest oscilator ca sursa pentru

ceasul de sistem este nevoie sa fie modificati bitii fuse. Bitii fuse sunt localizati intr-o zona rezervata din memoria EEPROM iar modificarea acestora se poate face numai cu un programator dedicat deoarece bootloaderul nu poate accesa acel sector al memoriei. Bitii fuse sunt o serie de biti ce ajuta la configurarea microcontrolerului, aceastia formeaza impreuna 3 bytes: High Fuse Byte, Low Fuse Byte si Extended Fuse Byte, pentru fiecare bit "1" inseamna programat iar "0" inseamna neprogramat. Pentru a fi siguri ca am programat corect acesti biti am folosit resursa https://www.engbedded.com/fusecalc/, unde am ales tipul de oscilator in conformitate cu foaia de catalog a microcontrolerului adica Full Swing Crystal Oscilator cu timp de pornire cat mai mare ca sa asigure o durata suficienta stabilizarii oscilatiilor si am dezactivat optiunea ce imparte ceasul la 8. Modificarea bitilor fuse a fost efectuata cu programul avrdude folosind comanda "avrdude -c usbasp -p m164p -U lfuse:w:0xe7:m -U hfuse:w:0x99:m". Pentru a ne asigura ca microcontrolerul foloseste intradevar frecventa de 20 MHz, am definit in cod frecventa ceasului la un 1 MHz si am tinut aprins un LED timp de 2 secunde. In acest caz frecventa ceasului intern fiind de 1 MHz, LED-ul a stat aprins pentru 2 secunde, exact cat a fost programat(..insert link here...). Am cuplat apoi oscilatorul si am rulat acelasi program, in acest caz LED-ul nu mai sta aprins timp de 2 secunde, ci mult mai putin(...insert linke here...). In cod am definit frecventa de 1 MHz iar microcontrolerul presupune ca aceasta este frecventa ceasului intern si calculeaza in functie de ea timpul pentru care tine LED-ul aprins, iar apoi pentru a-l tine aprins utlizeaza ceasul intern, de aceea chiar daca calculul s-a facut pentru o frecventa de 1 MHz, LED-ul nu va sta aprins pentru 2 secunde deoarece frecventa ceasului este mult mai mare, in acest caz 20 MHz iar timpul in care sta LED-ul aprins este de 0.1 secunde. In urma acestui test putem spune cu exactitate ca microcontrolerul seteaza ceasul intern dupa oscilatorul cuplat.

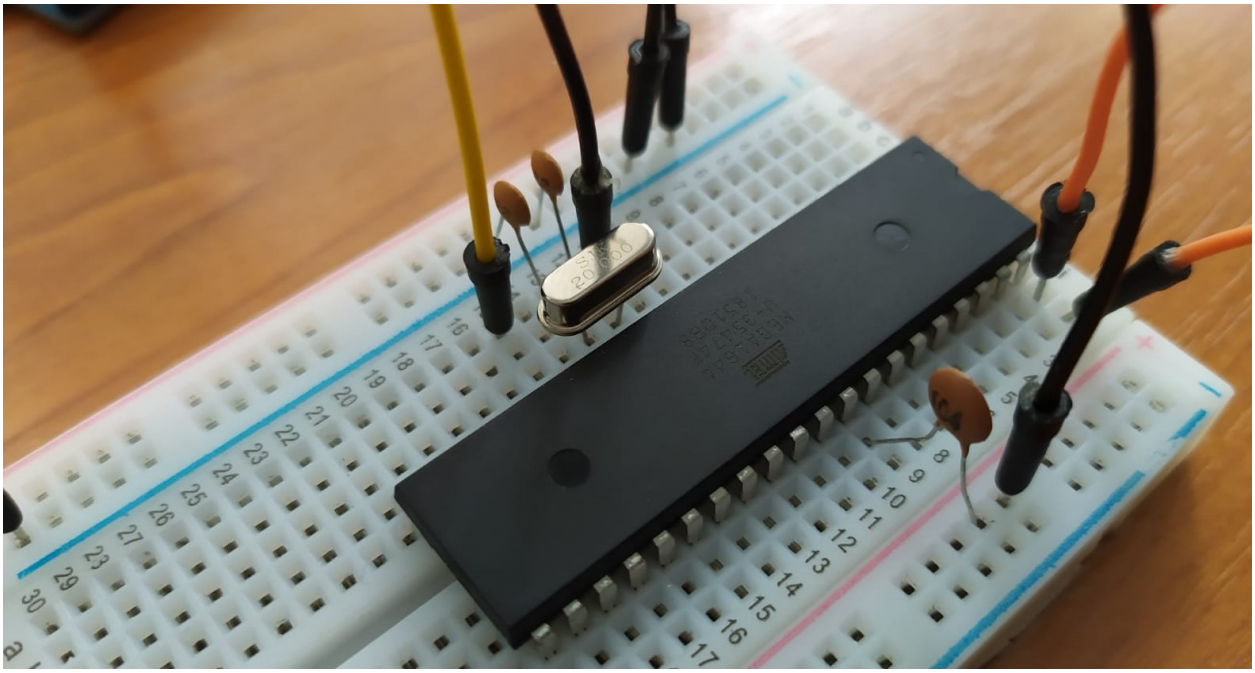

Figure 3: Oscilatorul cuplat la uC conectat la pinii XTAL1 si XTAL2 cu 2 condensatoare de 22 pF, valori extrase din foaia de catalog.

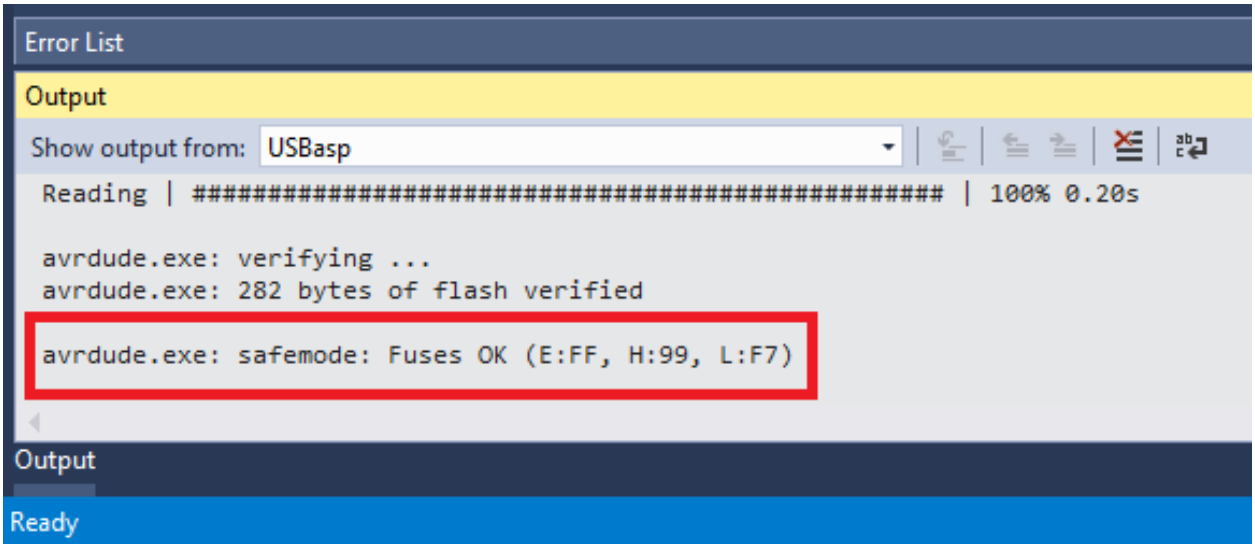

Figure 4: Programarea bitilor fuse cu avrdude USB ASP

6 Comunicatia seriala

Microcontrolerul este capabil de a comunica cu alte dispozitive, aceasta comunicatie asigurandu-se printr-o multitudine de protocoale si interfete, insa ptrintre acestea nu se afla si USB; de aceea pentru a realiza comunicatia intre uC si PC prin intermediul portului USB este nevoie de un convertor USB-TTL. Rolul acestuia este de a adapta interfata folosita de microcontroler pentru comunicatie la cea folosita de PC pentru a se putea efectua schimbul de informatii. Pe tot parcursul desfasurarii proiectului, din cauza lipsei stocului nu am putut face rost de un convertor USB-TTL dedicat, insa am avut la dispozitie o placa de dezvoltare Arduino ce are integrat un astfel de convertor si poate fi configurata sa functioneze si cu alte dispozitive in afara de uC sau integrat. Legand pinul RESET la GND pe placa de dezvoltare Arduino putem suspenda microcontrolerul integrat al placii si folosi convertorul USB-TTL pentru a comunica cu alte dispozitive prin intermediul pinilor RX si TX. Optional, folosind aceasta placa de dezvoltare se poate asigura alimentarea microcontrolerului.

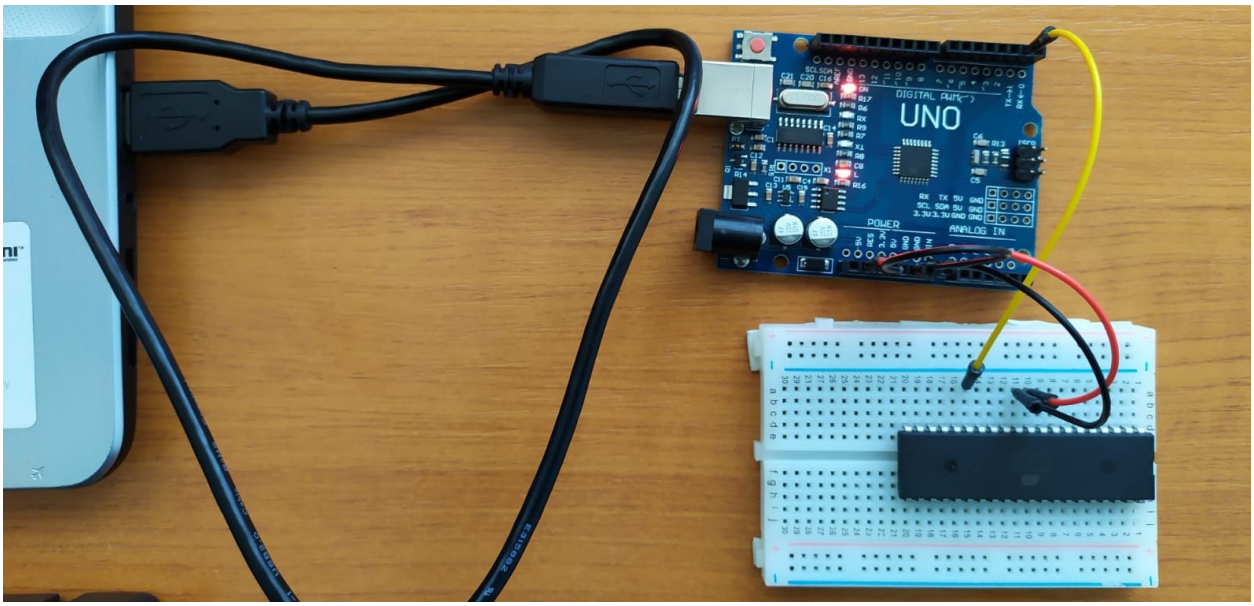

Figure 5: Placa de dezvoltare Arduino folosita pe post de convertor USB-TTL

ATmega 164A dispune de urmatorele moduri de comunicatie seriala: SPI, USART, USART in modul SPI si TWI. Din cauza familiaritatii, usurin-

tei de implementare si simplitatii am ales sa folosim interfata USART, iar din cele doua prezente in functionalitatea microcontrolerului USART0 si USART1 vom folosi USART0. Pentru o mai mare flexibilitate am ales sa folosim pentru transmiterea si receptionarea datelor modul asincron de lucru, deci se impune setarea unei rate de baud; aceasta este calculata folosind frecventa interna a ceasului si continutul registrului UBBRn. Practic pentru a seta rata de baud este necesara scrierea unei valori in registrul UBBRn, aceasta valoare este deci calculata dupa formula $UBBRn = \frac{f_{OSC}}{16BAUD} - 1$. Datele vor fi transmise in aceasta comunicatie in formate. Un format de date suporta 1 bit de start, 5 pana la 9 biti de date, 1 bit de paritate si unul sau doi biti de stop. Linia de comunicatie ramane la nivelul logic de 1 in starea de asteptare dupa care schimbarea nivelului catre nivelul de 0 logic semnifica aparitia bitului de start, apoi sunt transmisi cei 5 pana la 9 biti de date, bitul de paritate optional si bitii de stop, dupa care linia ramane din nou in asteptare, stare corespunzatoare nivelului de 1 logic pana la urmatoarea transmisie a altui format de date. Deoarece dimensiunea datelor cu care lucram se poate exprima in functie de numarul de octeti pe care il ocupa in memorie, cea mai convenabila alegere este cea de 8 biti de date. In aceasta comunicatie sursele de interferenta sunt minime, astfel ca am deci transmisia fara bitul de paritate iar pentru simplitate ne vom rezuma la un bit de stop.

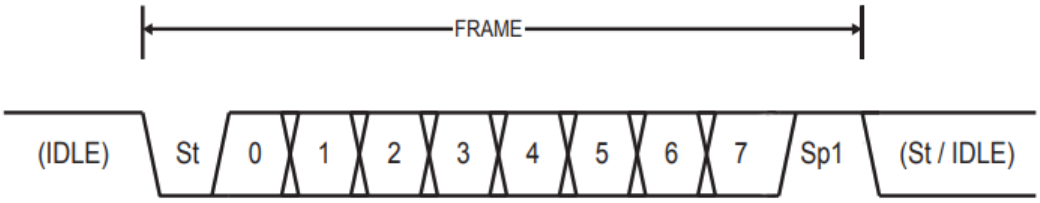

Figure 6: Formatul de date ales in cadrul acestui proiect

Setarea tuturor parametrilor dar si controlul transmisiei si receptiei se efectueaza cu ajutorul a 3 registre speciale de stare si control: UCSR0A, UCSR0B si UCSR0C. Deoarece timpul nu este o variabila critica in acest proiect am ales o rata de baud de 9600 simboluri/s. Pentru a activa transmisia si receptia se seteaza bitii TXEN0 si respectiv RXEN0 din registrul

UCSR0B. Arhitectura AVR a acestui microcontroler ne pune la dispozitie registrul UDR0 pentru schimbul de informatii, registru de deplasare pe 8 biti. In momentul in care acest registru este scris si daca transmisia este activata se va incepe transmiterea lui pe interfata seriala, iar daca receptia este activata, orice date receptionate se vor gasi in acest registru. Desi denumirea este comuna atat pentru receptie cat si pentru transmisie, modul in care este folosit (daca se scrie sau se citeste) determina care registru este adresat (existand astfel 2 registre cu numele de UDR0), USART facilitand o comunicatie full duplex (se poate transmite si receptiona independent).

7 Etalonarea senzorilor

De obicei orice componenta electronica activa sau de o compelxitate ridicata este livrata impreuna cu o foaie de catalog de la producator care descrie in detaliu functionarea acesteia. Din cauza bugetului redus am achizitionat senzori ce nu dispun de foi de catalog, ci numai de o serie de parametrii listati de distribuitor. Chiar daca majoritatea senzorilor au o caracteristica ideala liniara am decis ca este necesara etalonarea pentru a garanta functionarea acestor in anumite limite. Ambii senzori, atat cel de nivel cat si cel de umiditate a solului ar trebui ideal sa aiba caracteristici liniare. Senzorii au o marime de intrare reprezentata de marimea fizica ce trebuie masurata si o marime de iesire, pentru ambii senzori reprezentata de o tensiune electrica ce va trebui citita. Pentru a etalona senzorii vom privi ambele marimi ca doua variabile aleatoare, vom prelua date empirice si vom folosi metode statistice pentru a estima parametrii de interes. Vom nota astfel cu X marimea de intrare si cu Y marimea de iesire si vom incerca estimarea liniara a variabilei Y cu ajutorul variabilei X determinand o functie de forma $\hat{Y} = \varphi(X)$. Pentru a determina aceasta functie vom utiliza metoda celor mai mici patrate prin care ne dorim ca $M[(Y - \varphi(X))^2] = min$. In cazul nostru cea mai buna estimare este una de tip liniar si neomogen astfel ca $\hat{Y}$ va fi de forma $\hat{Y} = aX + b$, $a, b \in \Re$. Microcontrolerul va citi variabila Y si va incerca sa estimeze valoarea lui X de aceea dupa determinarea dependentei intre X si Y vom determina inversa functiei adica φ^{-1}. Pentru ambii senzori am ales sa prelucram datele folosind mediul Matlab si am determinat caracteristica trasand un polinom

de gradul unu ce minimizeaza media patratului erorii $\varepsilon = Y - \varphi(X)$. In Matlab ne-au fost returnati coeficientii $a, b \in \Re$ ai acestui polinom.

7.1 Senzorul de nivel al apei

X = nivelul apei [mm]
Y = tensiunea de iesire [V]
Pentru acest senzor am efectuat 3 masuratori independente, 2 cu senzorul alimentat la 5V cu Vcc mediu de 4.7857V si una la 3V cu Vcc mediu de 3.24V, pentru ultima masuratoare senzorul a prezentat o caracteristica puternic neliniara si nu a fost inclusa in calcule, in acelasi timp ne poate sugera ca functionarea senzorului este mai stabila pentru alimentare la 5V. Pentru a determina valorile coeficientilor am conclus ca media primelor doua masuratori este mai apropiata de valoarea reala.
Din datele masurate am putut extrage $a = 0.02795, b = 2.044 \Rightarrow \hat{Y} \approx 0.03X + 2$, de unde $\varphi^{-1}(x) = \frac{100x-200}{3}$, deci am obtinut functia ce face legatura intre tensiunea citita si nivelul apei din rezervor.

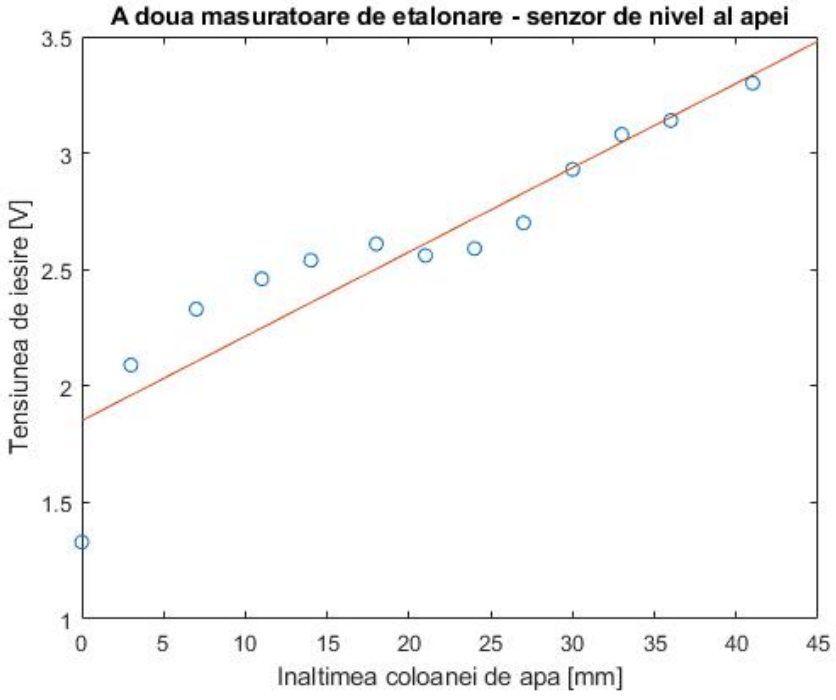

Figure 7: Determinarea caracteristicii senzorului de nivel

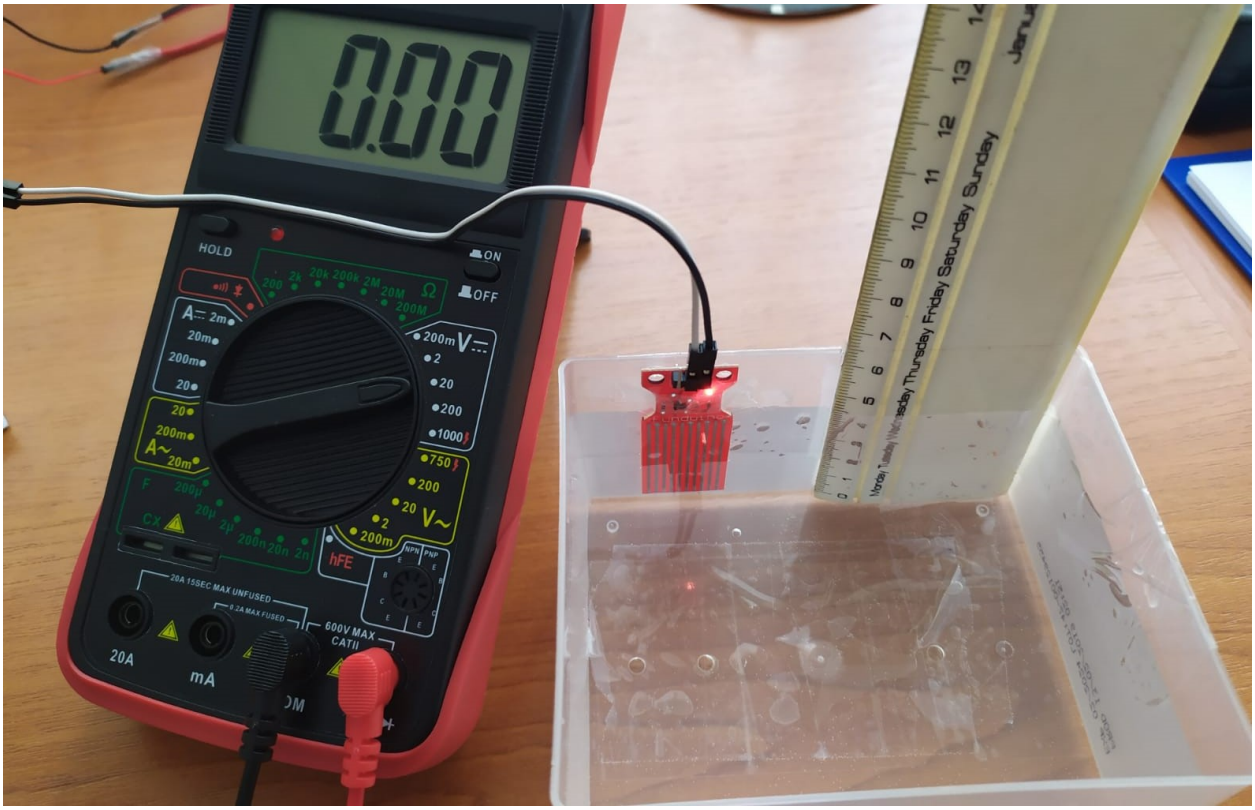

Figure 8: Masurarea empirica a marimii de iesire pentru senzorul de nivel

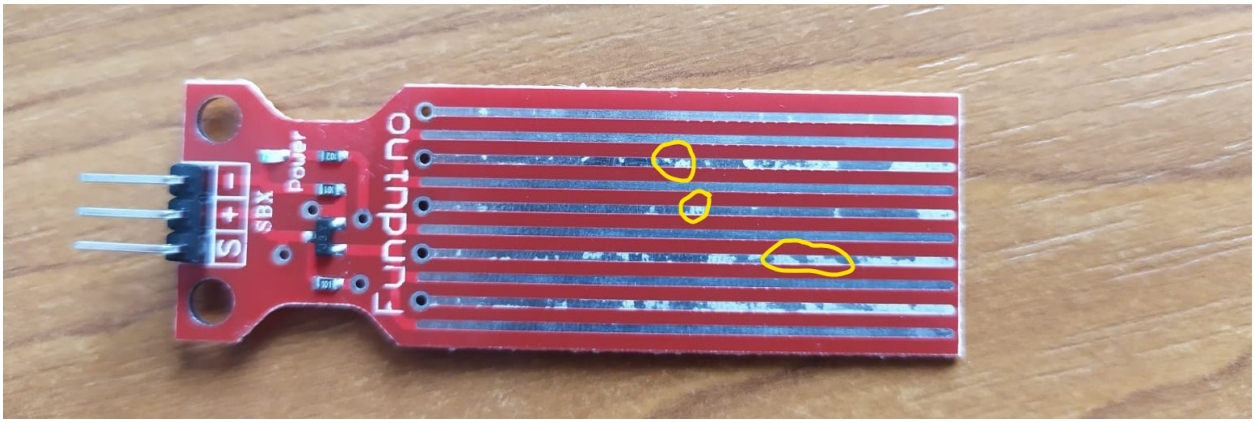

Figure 9: Coroziunea aparuta pe suprafata senzorului

Pentru ca sezorii achizitionati sunt relativ ieftini acestia nu sunt calitativi, se poate observa o consecinta a acestui fapt prin aparitia pe suprafata senzorului a unui tip de coroziune concentrata in pete. In timp, coroziunea poate afecta performantele senzorului.

7.2 Senzorul de umiditate a solului

X = umiditatea solului [%]
Y = tensiunea de iesire [V]
Am notat greutatea solului umed = gsum si greutatea solului uscat = gsu, astfel putem calcula umiditatea solului exprimata in procente dupa formula $umiditatea = \frac{gsum-gsu}{gsu} \cdot 100$ [%]
Pentru etalonarea senzorului de umiditate am ales o mostra de sol cu greutatea aproximativa de 100 g. Deoarece nu am avut acces la instrumente de precizie aceasta masuratoare este grosiera, ca atare si aproximarea caracteristicii senzorului va fi una grosiera.
Pentru acest senzor am efectuat 3 masuratori insa de data aceasta valorile au fost preluate una dupa cealalta, ceea ce nu ne mai da posibilitatea de a spune ca masuratorile sunt independente, pentru calculul parametrilor am considerat o singura masuratoare unde am mediat fiecare valoare.
Din datele masurate am putut extrage $a = -0.042, b = 4.7981 \Rightarrow \hat{Y} \approx -0.04X + 4.8$, de unde $\varphi^{-1}(x) = -5(5x - 24)$, deci am obtinut functia ce face legatura intre tensiunea citita si umiditatea solului.

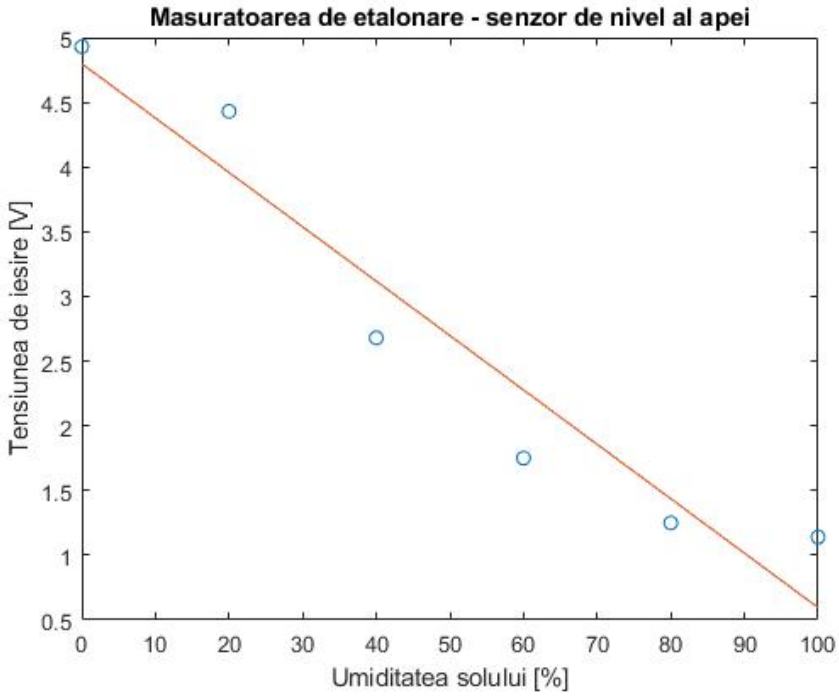

Figure 10: Determinarea carcateristicii senzorului de umiditate

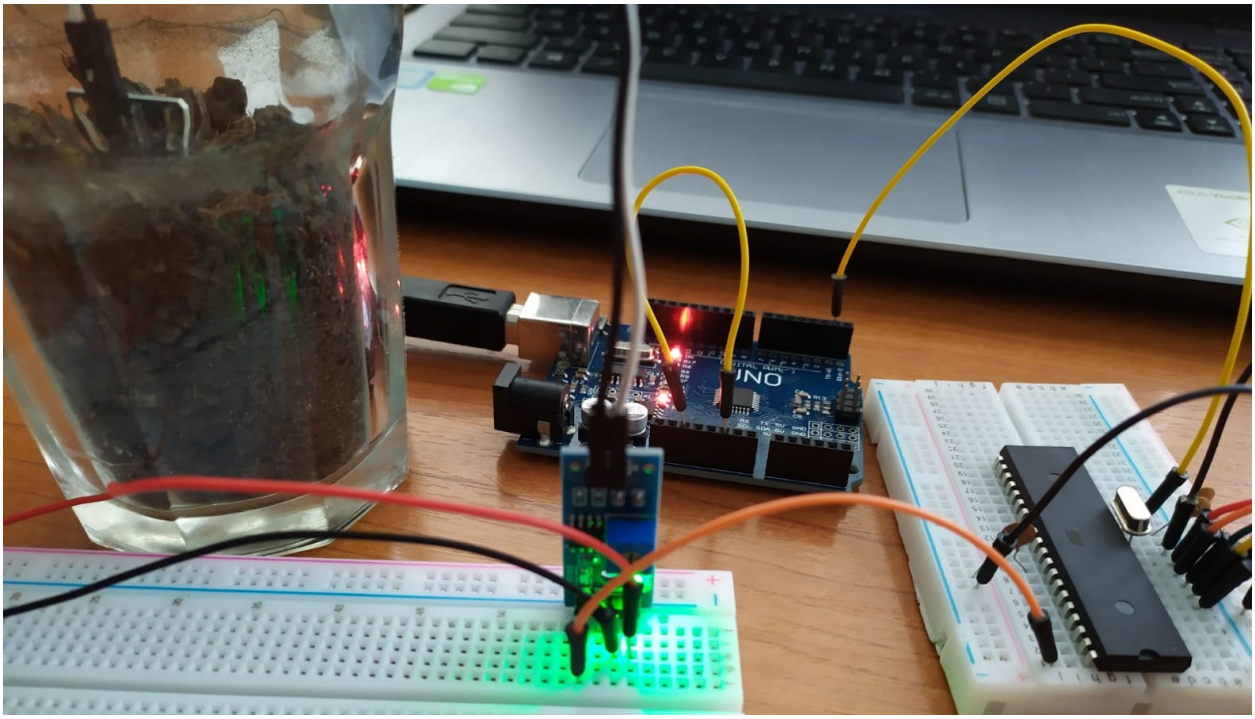

Figure 11: Masurarea empirica a marimii de iesire pentru senzorul de umiditate a solului

S-a incercat in paralel si masurarea cu ajutorul microcontrolerului, insa rezultatele furnizate de acesta erau inconsistente si de o deviatie mare fata de valorile date de multimetru, astfel ca nu s-au putut extrage informatii din aceasta masuratoare.

8 Actuatorul

Ca actuator pentru a extrage apa din rezervor si a o transporta catre ansamblul de plante am ales o minipompa de apa submersibila. De asemenea si in cazul acesteia din cauza bugetului am achizitionat o pompa ce nu a venit cu o foaie de catalog ci cu o serie de specificatii din partea distribuitorului. Din pacate pompa a ajuns nefunctionala, insa am reusit sa gasim problema si sa o reparam. Minipompa de apa consta dintr-un motor DC si un ansamblu mecanic format dintr-o paleta si o incapere cu 2 orificii, unul prin care apa intra in incapere si unul prin care iese directionata de paleta, ce este la randul sau pusa in miscare de rotatie de catre motor. Problema pentru pompa noastra a fost ca paleta se afla mult prea aproape de corp ceea ce determina

o forta de frecare mult prea mare pentru a fi dezvoltata de motor, si in acest mod chair daca era alimentata, pompa nu functiona. Solutia a constat in deplasarea paletei pe axul ce o conecteaza la motor astfel incat sa se reduca frecarea cu corpul pompei. In urma acestei actiuni am reusit sa facem pompa functionala, insa parametrii sai s-au schimbat. Debitul s-a redus de la 1.2-1.4 L/min la aproximativ 0.43 L/min iar consumul sau de curent este usor crescut de la 250-270 mA la aproximativ 300 mA.

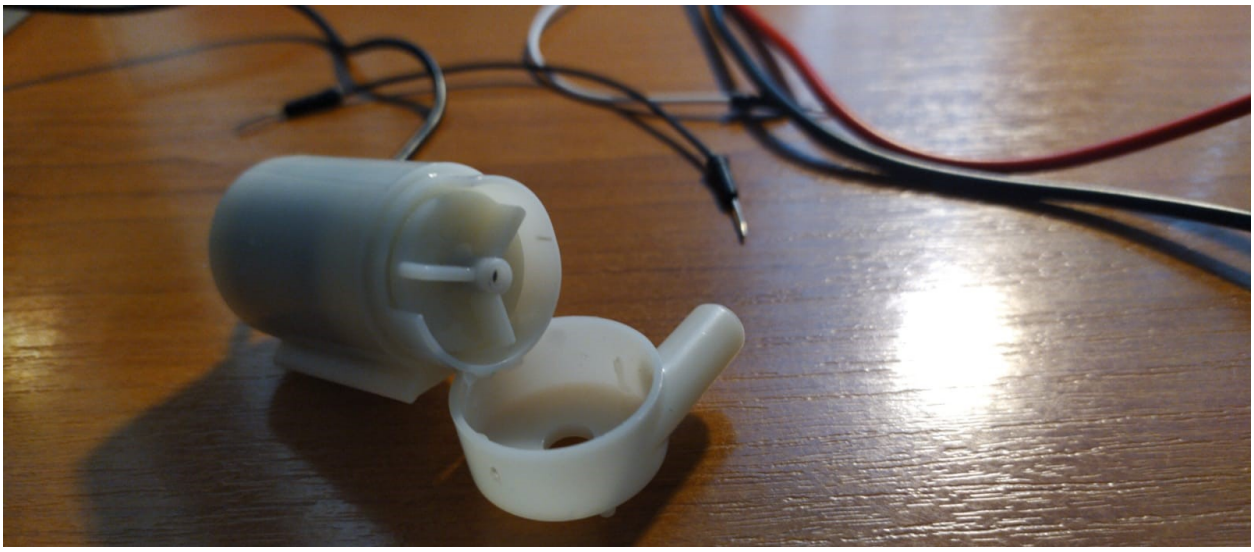

Figure 12: Pompa de apa, se poate observa paleta rotativa si incaperea cu cele 2 orificii

Repararea pompei a reprezentat un proces prin care i s-a distrus integritatea fizica insa in cea mai mare masura aceasta a putut fi restaurata. Consumul pompei este de $\approx$ 300 mA, iar curentul maxim ce poate fi furnizat de unul dintre pinii generali ai microcontrolerului este de 40 mA, ceea ce inseamna ca nu va fi posibila comanda actuatorului direct de catre microcontroler. In acest caz solutia aleasa de noi este comandarea pompei prin intermediul unui tranzistor ce va comuta intre saturatie ("deschis") si blocare ("inchis"). Pompa va fi amplasata astfel incat curentul consumat sa fie curentul de colector al tranzistorului.

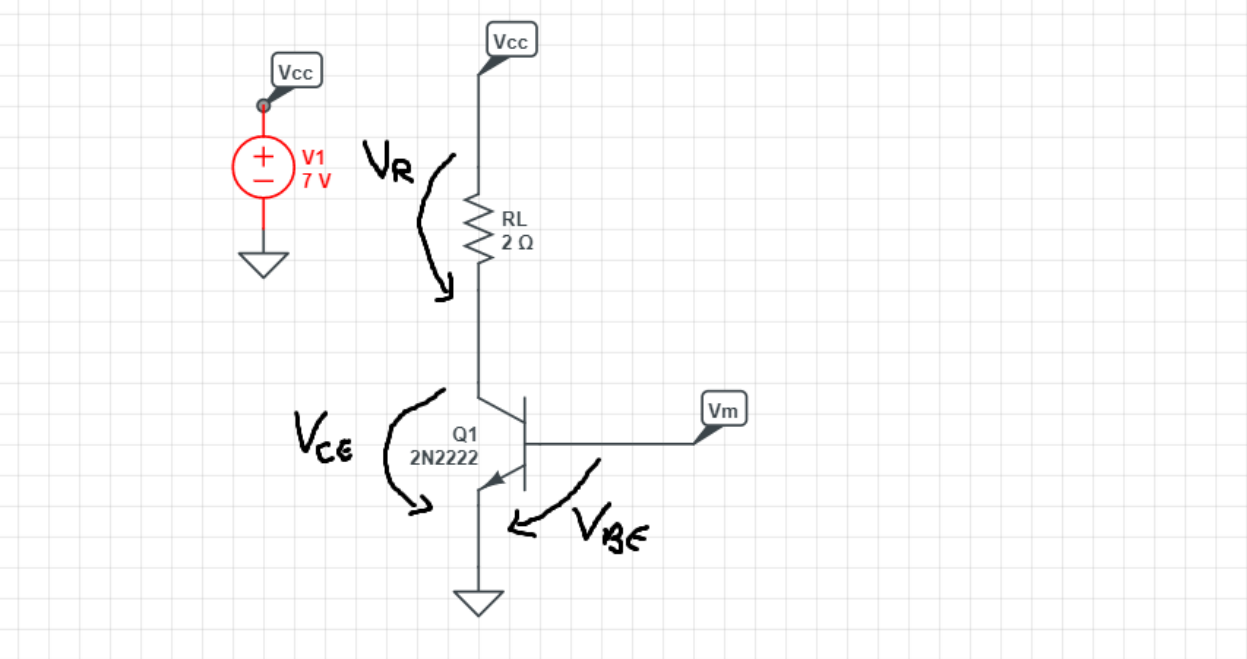

Figure 13: Schema prin care se doreste comandarea pompei de apa

Din cauza accesului limitat la piese, explicat in prima sectiune, am folosit cel mai adecvat tranzistor pe care l-am avut la dispozitie, acesta este tranzistorul bipolar NPN de tip 2N2222A. Pentru acest tranzistor am masurat un factor de amplificare static in curent $\beta \approx 300$. Vm reprezinta potentialul ce poate fi furnizat pe unul din pinii microcontrolerului, acesta avand la dispozitie pentru iesire numai pini digitali, deci Vm $\approx$ Vcc $\approx$ 5V. Tensiunea de functionare la care vom alimenta pompa este de 5V deci Vr trebuie sa fie minim 5V, pentru functionarea in saturatie s-a ales $V_{CE} \approx 2V$ ceea ce impune $V_{CC} \approx 7V$.

- $Vm = 0 \Rightarrow V_{BE} = 0; V_{CC} = 7V \Rightarrow V_{CE} > 0, V_{CB} > 0 \Rightarrow$ blocare.
- $Vm = 5V \Rightarrow V_{BE} > 0; V_{CC} = 7V \Rightarrow V_{CE} = 2V, V_R = 5V, V_{BC>0} \Rightarrow$ saturatie.

In regim de saturatie transzistorul ar trebiu sa prezinte o rezistenta foarte mica colector-emitor pentru a permite curegerea celor 300 mA consumati de pompa.

La implementarea practica a circuitului, la momentul aplicarii potentialului Vm, din motive pe care nu le cunoastem uC inceteaza sa functioneze, astfel circuitul nu a putut fi pus in practica. Avem ca solutie alternativa folosirea unui releu in locul tranzistorului.

9 Convertorul ADC

Pentru a putea lua orice decizie sau pentru a putea calcula parametrii de interes este nevoie de cuantizarea marimilor citite de la senzori si transformarea acestora sub o forma ce poate fi prelucrata numeric. Microcontrolerul dispune de un convertor ADC ce poate face acesata transformare. Convertorul ADC lucreaza pe 10 biti si poate avea ca referinta de tensiune: pinul extern AREF (referinta interna dezactivata), pinul AVCC cu condendsator cuplat la pinul AREF catre masa, referinta interna de 1.1V sau referinta interna de 2.56V. In acest proiect am ales ca referinta de tensiune pinul AVCC. Tot ce presupune controlul convertorului se face cu ajutorul registrelor ADCSRA si ADCSRB. Pentru a activa convertorul trebuie setat bitul ADEN din registrul ADCSRA iar pentru a porni o conversie bitul ADSC din acelasi registru. In momentul finalizarii unei conversii, microcontrolerul seteaza bitul ADSC inapoi la 0 iar rezultatul conversiei este scris in registrul de 16 biti ADC format din registrele de 8 biti ADCL si ADCH. Cand ADCL este citit, registrul de date ADC nu este actualizat pana nu se citeste si ADCH.

10 Schema electrica

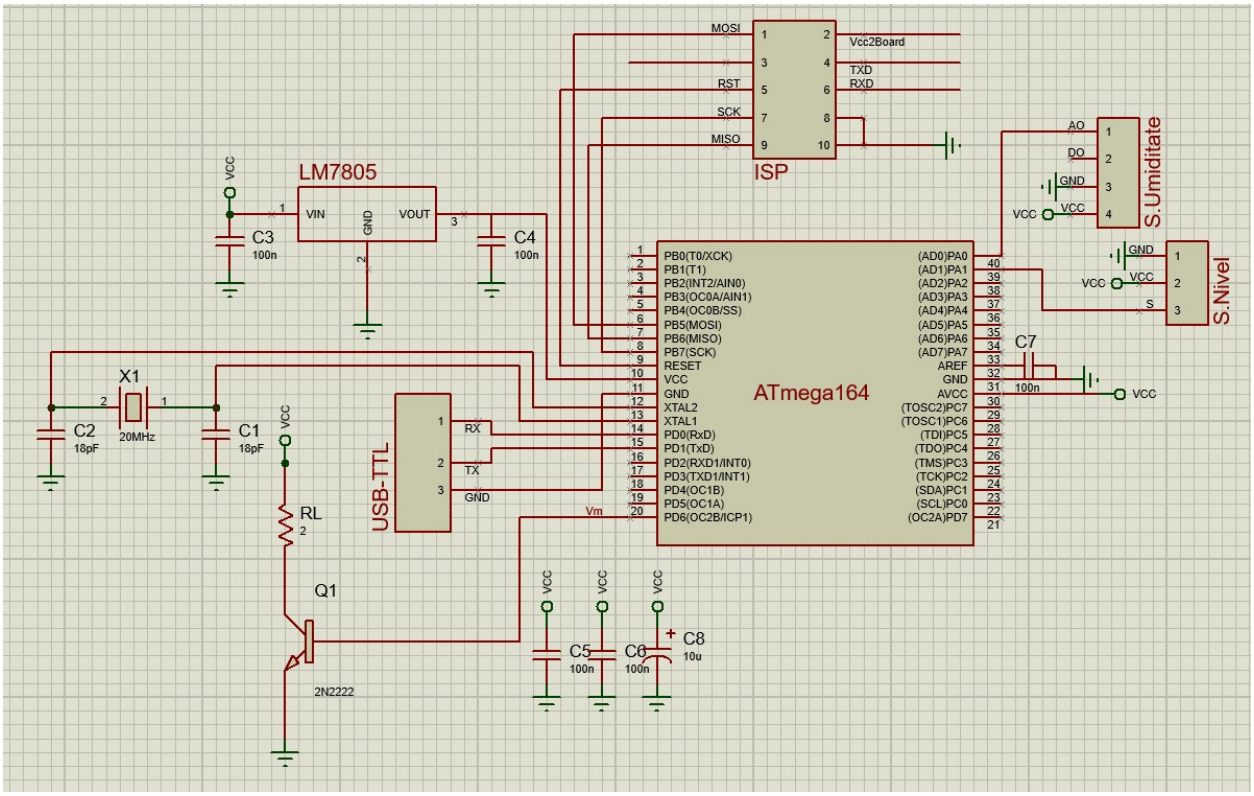

Figure 14: Schema electrica a proiectului

11 BOM (Bill of Materials)

Clasa	Descriere	Distribuitor	Cantitate	Pret unitar
Senzor	Senzor de nivel al apei	Optimus Digital	1	3,29 Lei
Senzor	Senzor de umiditate a solului	Optimus Digital	1	3,99 Lei
Actuator	Mini pompa de apa submersibila	Optimus Digital	1	8,99 Lei
Programator	Programator USB ASP	Optimus Digital	1	7,49 Lei
Stabilizator	Stabilizator de tensiune LM7805	ETTI	1	-
Convertor	Convertor USB TTL	Optimus Digital	1	12,89 Lei
Tranzistor	NPN 2N2222A	Optimus Digital	1	0,99 Lei
Oscilator	Quartz, 20 MHz	ETTI	1	-
Condensator	Electrolitic, 1uF	Optimus Digital	1	0,49 Lei
Condensator	Electrolitic, 10uF	Optimus Digital	1	0,49 Lei
Condensator	Ceramic, 22pF	Optimus Digital	2	0,49 Lei
Condensator	Ceramic, 100nF	Optimus Digital	5	0,69 Lei

12 Codul ce ruleaza pe microcontroler

```
1
2  #define F_CPU 20000000UL
3  #include <avr/io.h>
4  #include <util/delay.h>
5  #include <stdint.h>
6  #define MINUT 60000
7  #define VREF 5.1
8  #define DEBIT 0.43
9  #define BAUD 9600
10 #define MYUBRR F_CPU/16/BAUD-1
11
12 union {
13   float f;
14   char bytes[4];
15 } joinedFloat;
16
17 union {
18   int i;
19   char bytes[2];
20 } joinedInt;
21
22 float cantitateApa = -1; //L  // configurare numai prin aplicatie
23 int umiditatePrag = -1; //[%]  // configurare numai prin aplicatie
24
25 void ADC_Init();
26 void USART_Transmit( unsigned char data );
27 void USART_Init( unsigned int baud );
28 unsigned char USART_Receive( void );
29 void Conversie(unsigned char *low, unsigned char *high);
30 void stateConnected(unsigned char command);
31 void stateReading(unsigned char command);
32 int preiaDateUmiditate();
33 int preiaDateNivel();
34
35 int main(void)
36 {
37   //Initializare USART
38   USART_Init ( MYUBRR );
39
40   //Initializare ADC
41   ADC_Init();
42
43   //DDRD = 0xFF;
44   unsigned char command;
45   int umiditate; //[%]
46   int nivel; // [mm]
47
48     while (1)
49     {
```

```
50      command = UDR0;
51      if(command == 'c')
52        stateConnected(command);
53      umiditate = preiaDateUmiditate();
54      if(umiditatePrag != -1)
55        if(umiditate < umiditatePrag && cantitateApa != -1){
56          nivel = preiaDateNivel();
57          if(nivel > 0 && nivel <= 40){
58            float t = cantitateApa / DEBIT; //Calcul timpul in care se tine
    pompa deschisa
59            t *= 60; //Conversie in secunde
60            const int timp = t * 1000; //Conversie in ms
61            DDRD = 0xFF; //Setam portul D ca output
62            PORTD = 0xFF; //Setam potentialul Vm la aprox. Vcc = 5V
63            _delay_ms(1000); //Tinem pompa activa un anumit timp
    ----------------------------> PROBLEMA
64            PORTD = 0x00; //Inchidem pompa
65            _delay_ms(MINUT);//Asteptam un minut pentru infiltrarea apei in sol
66          }
67        }
68      }
69  }
70
71    //Returneaza umiditatea in procente
72  int preiaDateUmiditate(){
73
74    //Selecteaza canalul ADC0 pentru conversie (XOR cu 0x01)
75    ADMUX = ADMUX ^ 0x01;
76
77    ADCSRA = (ADCSRA | (1 << ADSC)); //Porneste o conversie setand ADSC
78    while((ADCSRA&(1<<ADIF))==0);    //Asteapta sa se termine conversia (ADSC
      resetat de hw)
79
80    //A doua conversie deoarece din cauza schimbarii canalului
81    //este posibil ca prima sa nu fie corecta
82    //ADCSRA = (ADCSRA | (1 << ADSC)); //Porneste o conversie setand ADSC
83    //while((ADCSRA&(1<<ADIF))==0);    //Asteapta sa se termine conversia (ADSC
      resetat de hw)
84
85    unsigned int convLow = ADCL; //Citim ADCL
86    unsigned int convHigh = (unsigned int)(ADCH << 8); //Citim ADCH si facem loc
        pentru ADCL shitand la stanga 8 pozitii
87    unsigned int conv = convLow | convHigh; //Concatenam ADCL cu ADCH
88
89    float resConv = (conv * VREF) / 1023.0; //Rezultatul conversiei in V
90    int umiditate = -5 * (5*resConv -24); //Umiditatea in procente (0-100)
91
92    return (umiditate / 10) * 10; //Discretizarea valorilor pe 10 intervale
      pentru a minimiza eroarea aprox. caracteristicii
93
94  }
```

```
95
96    //Returneaza nivelul din vas in mm
97  int preiaDateNivel(){
98
99    //Selecteaza canalul ADC1 pentru conversie (SAU cu 0x01)
100   ADMUX = ADMUX ^ 0x01;
101
102   ADCSRA = (ADCSRA | (1 << ADSC)); //Porneste o conversie setand ADSC
103   while((ADCSRA&(1<<ADIF))==0);    //Asteapta sa se termine conversia (ADSC
        resetat de hw)
104
105   //A doua conversie deoarece din cauza schimbarii canalului
106   //este posibil ca prima sa nu fie corecta
107   //ADCSRA = (ADCSRA | (1 << ADSC)); //Porneste o conversie setand ADSC
108   //while((ADCSRA&(1<<ADIF))==0);    //Asteapta sa se termine conversia (ADSC
        resetat de hw)
109
110   unsigned int convLow = ADCL; //Citim ADCL
111   unsigned int convHigh = (unsigned int)(ADCH << 8); //Citim ADCH si facem loc
        pentru ADCL shitand la stanga 8 pozitii
112   unsigned int conv = convLow | convHigh; //Concatenam ADCL cu ADCH
113
114   float resConv = (conv * VREF) / 1023.0; //Rezultatul conversiei in V
115   int nivel = (100 * resConv - 200) / 3; //Nivelul in mm (0-40)
116
117   return nivel;
118
119 }
120
121 void stateConnected(unsigned char command){
122
123   int OK = 0;
124
125   while(command == 'c'){
126
127     if(!OK){
128       USART_Transmit('a'); //Transmitere mesaj de confirmare intrare in starea
        de conectat
129       OK = 1;
130     }
131     int umiditate = preiaDateUmiditate(); //Se preia umiditatea
132     int nivel = preiaDateNivel(); //Se preia nivelul apei
133     unsigned char *p;
134     p = (unsigned char*)&umiditate;
135     USART_Transmit(*p++);
136     USART_Transmit(*p);
137     p = (unsigned char*)&nivel;
138     USART_Transmit(*p++);
139     USART_Transmit(*p);
140
141     _delay_ms(2000);
```

```
142
143
144       command = UDR0;
145       if(command == 'r'){
146         stateReading(command);
147         command = UDR0;
148         command = 'c';
149       }
150       else if(command != 'b')
151         command = 'c';
152
153     }
154   }
155
156   void stateReading(unsigned char command){
157
158     int i;
159     for(i=0;i<4;i++){
160       joinedFloat.bytes[3-i] = USART_Receive();
161     }
162
163     cantitateApa = joinedFloat.f;
164
165     for(i=0;i<2;i++){
166       joinedInt.bytes[1-i] = USART_Receive();
167     }
168
169     umiditatePrag = joinedInt.i;
170   }
171
172
173   void ADC_Init()
174   {
175     //Activeaza ADC, selecteaza intreruperile si
176     //defineste frecventa de lucru a ADC la XTAL/2
177     ADCSRA = 0x87;
178
179     //Selecteaza Vref ca AVCC, rezultatul ajustat la dreapta
180     //si canalul analog de intrare ca ADC0
181     ADMUX = 0x40;
182   }
183
184   void USART_Init( unsigned int baud )
185   {
186     /* Set baud rate */
187     UBRR0H = (unsigned char)(baud>>8);
188     UBRR0L = (unsigned char)baud;
189     /* Enable receiver and transmitter */
190     UCSR0B = (1<<RXEN0)|(1<<TXEN0);
191     /* Set frame format: 8data, 2stop bit */
192     UCSR0C = (1<<USBS0)|(3<<UCSZ00);
```

```
193 }
194
195 void USART_Transmit( unsigned char data )
196 {
197   /* Wait for empty transmit buffer */
198   while ( !( UCSR0A & (1<<UDRE0)) )
199   ;
200   /* Put data into buffer, sends the data */
201   UDR0 = data;
202 }
203
204 unsigned char USART_Receive( void )
205 {
206   /* Wait for data to be received */
207   while ( !(UCSR0A & (1<<RXC0)) )
208   ;
209   /* Get and return received data from buffer */
210   return UDR0;
211 }
```

13 Codul aplicatiei de utilizator

```
1
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel;
5  using System.Data;
6  using System.Drawing;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace P2
13 {
14     public partial class Form1 : Form
15     {
16         bool isConnected = false;
17         public Form1()
18         {
19             InitializeComponent();
20             stateLabel.Text = "Neconectat";
21             buttonDeconnect.Enabled = false;
22             labelUmidVal.Text = "N/A";
23             labelNivelVal.Text = "N/A";
24             textBoxUmid.Text = "0";
25             textBoxCantApa.Text = "0";
26             textBoxUmid.Enabled = false;
27             textBoxCantApa.Enabled = false;
28             buttonSend.Enabled = false;
29         }
30
31         private void Form1_Load(object sender, EventArgs e)
32         {
33             comBox.Items.Add("COM1");
34             comBox.Items.Add("COM2");
35             comBox.Items.Add("COM3");
36             comBox.Items.Add("COM4");
37             comBox.Items.Add("COM5");
38             comBox.SelectedIndex = 0;
39             SerialPort.PortName = comBox.SelectedItem.ToString();
40         }
41
42         private void buttonConnect_Click(object sender, EventArgs e)
43         {
44             buttonConnect.Enabled = false;
45             buttonDeconnect.Enabled = true;
46             textBoxUmid.Enabled = true;
47             textBoxCantApa.Enabled = true;
48             buttonSend.Enabled = true;
49
```

```
50              SerialPort.Open();
51              SerialPort.WriteLine("c");
52              char ack = (char)SerialPort.ReadChar();
53              if(ack == 'a')
54              {
55                  isConnected = true;
56                      byte[] b = new byte[2];
57                      b[0] = (byte)SerialPort.ReadByte();
58                      b[1] = (byte)SerialPort.ReadByte();
59                      int realtimevalue = b[1];
60                      realtimevalue = realtimevalue << 8;
61                      realtimevalue += b[0];
62                      labelUmidVal.Text = realtimevalue.ToString() + " %";
63
64                  b[0] = (byte)SerialPort.ReadByte();
65                      b[1] = (byte)SerialPort.ReadByte();
66                      realtimevalue = b[1];
67                      realtimevalue = realtimevalue << 8;
68                      realtimevalue += b[0];
69                  labelNivelVal.Text = realtimevalue.ToString() + " mm";
70              }
71          }
72
73          private void comBox_SelectedIndexChanged(object sender, EventArgs e)
74          {
75              SerialPort.PortName = comBox.SelectedItem.ToString();
76          }
77
78          private void buttonDeconnect_Click(object sender, EventArgs e)
79          {
80              SerialPort.WriteLine("b");
81              buttonConnect.Enabled = true;
82              buttonDeconnect.Enabled = false;
83              textBoxUmid.Enabled = false;
84              textBoxCantApa.Enabled = false;
85              buttonSend.Enabled = false;
86          }
87
88          private void buttonUpdate_Click(object sender, EventArgs e)
89          {
90              buttonUpdate.Enabled = false;
91              if (SerialPort.IsOpen && isConnected)
92              {
93                  byte[] b = new byte[2];
94                  b[0] = (byte)SerialPort.ReadByte();
95                  b[1] = (byte)SerialPort.ReadByte();
96                  int realtimevalue = b[1];
97                  realtimevalue = realtimevalue << 8;
98                  realtimevalue += b[0];
99                  labelUmidVal.Text = realtimevalue.ToString() + " %";
100
```

```
                b[0] = (byte)SerialPort.ReadByte();
                b[1] = (byte)SerialPort.ReadByte();
                realtimevalue = b[1];
                realtimevalue = realtimevalue << 8;
                realtimevalue += b[0];
                labelNivelVal.Text = realtimevalue.ToString() + " mm";

            }
            buttonUpdate.Enabled = true;
        }

        private void buttonSend_Click(object sender, EventArgs e)
        {

            if(textBoxUmid.Text != "" && textBoxCantApa.Text != "")
            {
                float cantitateApa = float.Parse(textBoxCantApa.Text);
                SerialPort.WriteLine("r");
                isConnected = false;
                byte[] b = BitConverter.GetBytes(cantitateApa);
                SerialPort.WriteLine(b[0].ToString());
                SerialPort.WriteLine(b[1].ToString());
                SerialPort.WriteLine(b[2].ToString());
                SerialPort.WriteLine(b[3].ToString());

                int umiditatePrag = int.Parse(textBoxUmid.Text);
                byte[] bi = BitConverter.GetBytes(umiditatePrag);
                SerialPort.WriteLine(bi[0].ToString());
                SerialPort.WriteLine(bi[1].ToString());

                SerialPort.WriteLine("x");
                isConnected = true;
            }
        }

        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            SerialPort.WriteLine("b");
            SerialPort.Close();
        }
    }
}
```

14 Bibliografie

- https://www.youtube.com/watch?v=IyGwvGzrqp8& list=WL& index=8& t=0s
- https://www.youtube.com/watch?v=F7NlCaaL3yU& list=WL& index=8& t=0s
- https://www.instructables.com/id/Serial-Port-Programming-With-NET/
- https://www.edn.com/usart-vs-uart-know-the-difference/
- https://www.allaboutcircuits.com/textbook/digital/chpt-3/logic-signal-voltage-levels/
- https://www.edaphic.com.au/soil-water-compendium/soil-moisture-sensor-calibration/
- https://electronics.stackexchange.com/questions/112441/using-a-transistor-as-a-switch
- https://www.instructables.com/id/Avr-fuse-basics-Running-an-avr-with-an-external-cl/
- https://www.engbedded.com/fusecalc/
- https://app.diagrams.net/
- https://www.avrfreaks.net/forum/converting-4-bytes-float-help