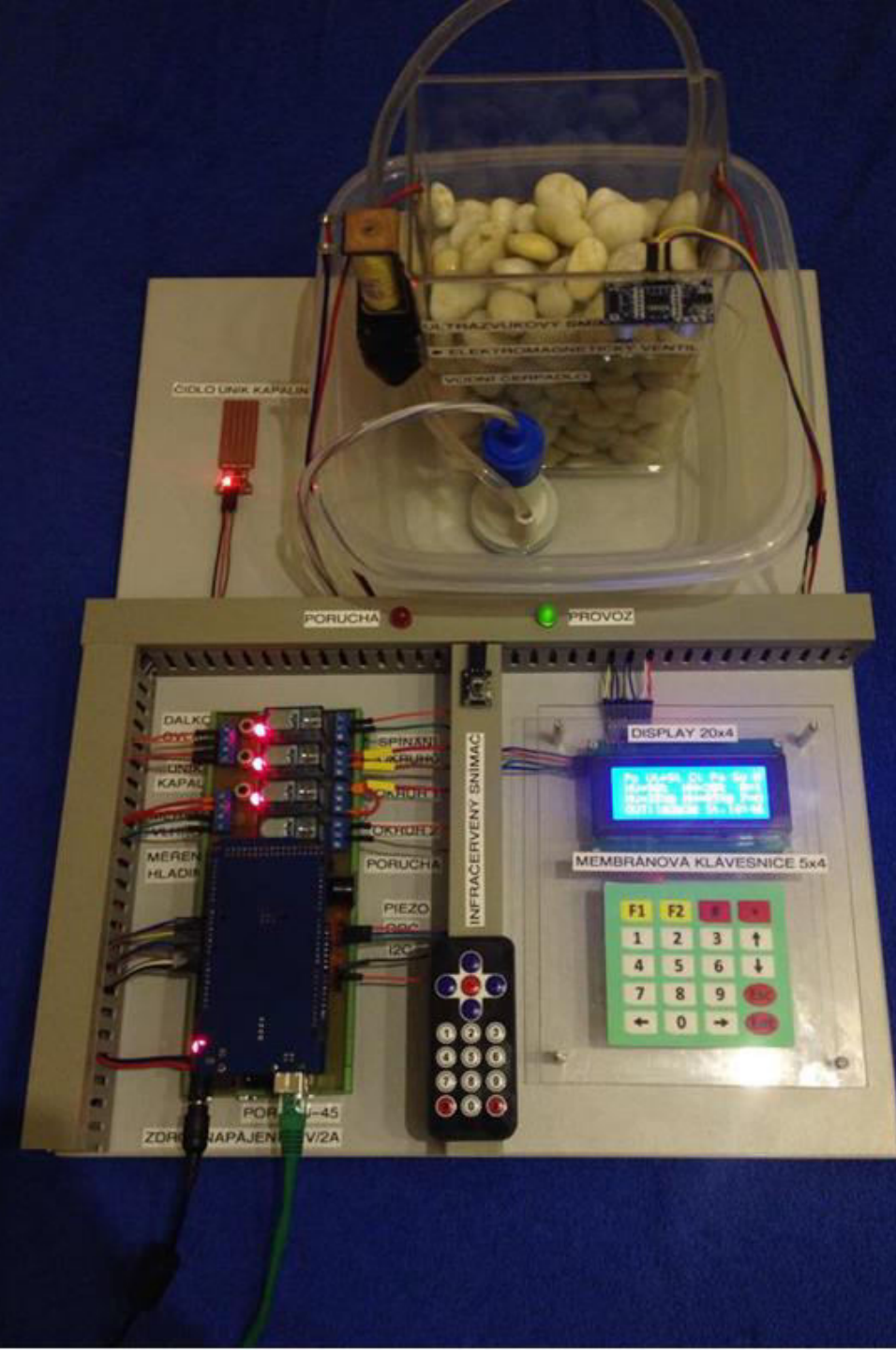


# Automatický zavlažovací systém

*Práci vytvořil Kočica Filip*  
*Konzultant PaedDr. Mahdal Antonín*



# Automatické zavlažovací systémy

- V současné době se stávají již samozřejmou součástí nejen soukromých okrasných zahrad a trávníků, ale i komerčních prostranství, jako jsou golfové a fotbalové hřiště, parky apod.
- Za stěžejní účel a největší výhodu lze považovat ušetření velkého množství času a práce, které by muselo být investováno pro pěkný trávník a zahradu.
- V neposlední řadě jsou estetičtější než zahradní hadice.



# Cíl práce

- Cílem této práce bylo vytvořit program pro elektronickou řídicí jednotku Arduino Mega 2560, zrealizovat možnost dálkového přístupu alias ethernetové rozhraní a také navrhnout a zrealizovat fyzické řešení plošného spoje, simulaci zavlažování, měření vlhkosti zavlažovaného prostředí, výšky hladiny vody v nádrži a poruchy systému.
- Na začátku tvorby systému jsem si stanovil 7 cílů, které vyplynuly z požadavků uživatelů z řad široké veřejnosti na rychlou, pohodlnou a jednoduchou manipulaci se systémem.



# Řízení systému

- Lokálně pomocí displeje 20 × 4 s membránovou klávesnicí 5 × 4 nebo s infračerveným dálkovým ovladačem.
- Vzdáleně pomocí ethernetového rozhraní.



Ethernetové rozhraní pro zavlažovací systém, Kočica Filip ME4						
PONDĚLÍ	ÚTERÝ	STŘEDA	ČTVRTEK	PÁTEK	SOBOTA	NEDĚLE
Nastavení parametrů						Režim + Kalibrace
Vstupy						Porucha rozpoznuta
Výstupy				Aktualní čas		



# Softwarové vybavení řídicí jednotky

- Struktura programu je vytvořena hlavně v závislosti na tom, že je potřeba provádět naplánované procesy a také celý systém vůbec musí pracovat v reálném čase (spouštět závlahu v přesně nastavený čas, měřit všechny parametry, číst a zapisovat data na displej i ethernet) nezávisle na stavu systému a ovládacích rozhraní.
- Program je rozdělen na hlavní funkce, které se starají o jednotlivé procesy.
- Více o řídicím softwaru se dozvíte v technickém koutku, kde jsem pro Vás připravil ukázkou zdrojových kódů nejdůležitějších procesů doplněnou o popis.

# Rozložení dat v hlavním menu

- Při rozmisťování dat na display byl kladen důraz na co nejvyšší čitelnost a srozumitelnost i pro technicky nezasvěcené osoby.
- Pohybovat v menu se lze kurzorem ve tvaru šipky.
- Na prvním řádku jsou dny Pondělí až Neděle.
- Na druhém řádku se zobrazují námi nastavené parametry a aktuální režim
- Na třetím se zobrazují naměřené hodnoty a porucha.
- Na čtvrtém jsou zobrazeny výstupy a aktuální den + čas.



•Po Ut St Ct Pa So N  
NU=90% NH=25% R=1  
MU=0%■ MH=100%■P=0  
OUT:1■203■ Ct, 22:07

# Základní funkce systému

Nastavení zavlažovacích dní

```
15:51 Nast. Hodiny
22:07 Ctvrtek
      05      01
Soucasny beh - Vyp
```

Měření průtoku

```
1/m: 1
0.00594 m³
0.04194 m³
0.06606 m³
```

Nastavení parametrů

```
*NAST. VLHKOSTI= 90%
NAST. HLADINY = 25%
NAST. HYSTEREZ= 1%
LIMIT DISPLAVE= 1m
```

Výběr režimu

```
* NASTAVENI REZIMU *
*Rezim NORMAL      (A)
Rezim SUCHO
Rezim VLHKO
```

Kalibrace

```
*KALIBRACE VLHKOSTI*
* KALIBRUJ!
*KALIBRACE HLADINY *
KALIBRUJ!
```

Manuální spouštění výstupů

```
*NASTAVENI VYSTUPU*
OUT1=■ AKTIVNI
*OUT2=■ AKTIVNI
OUT3=□ NEAKTIVNI
```

# Ethernetové rozhraní

## Ethernetové rozhraní pro zavlažovací systém, Kočica Filip ME4

PONDĚLÍ	ÚTERÝ	STŘEDA	ČTVRTEK	PÁTEK	SOBOTA	NEDĚLE
Nastavení parametrů					Režim + Kalibrace	
Vstupy					Porucha rozepnuta	
Výstupy				Aktualní čas		

Zpět

### Pondělí

0 8 : 4 4 - 10 : 20

### 1.Okruh

4 6 - 15

### 2.Okruh

9 2 - 30

Současný Běh je **VYPNUTÝ**

☒ ON  
☐ OFF

Submit



# Ethernetové rozhraní

## ➤ Nastavování číselných hodnot

0 8 : 4 4 -  :

1.0

! Value must be greater than or equal to 0.

0 8 : 4 4 -  :

1.0

! Value must be less than or equal to 23.

## ➤ Nastavování nečíselných hodnot

OUT 3 je ZAPNUTÝ

☐ ON

☐ OFF

☐ Po

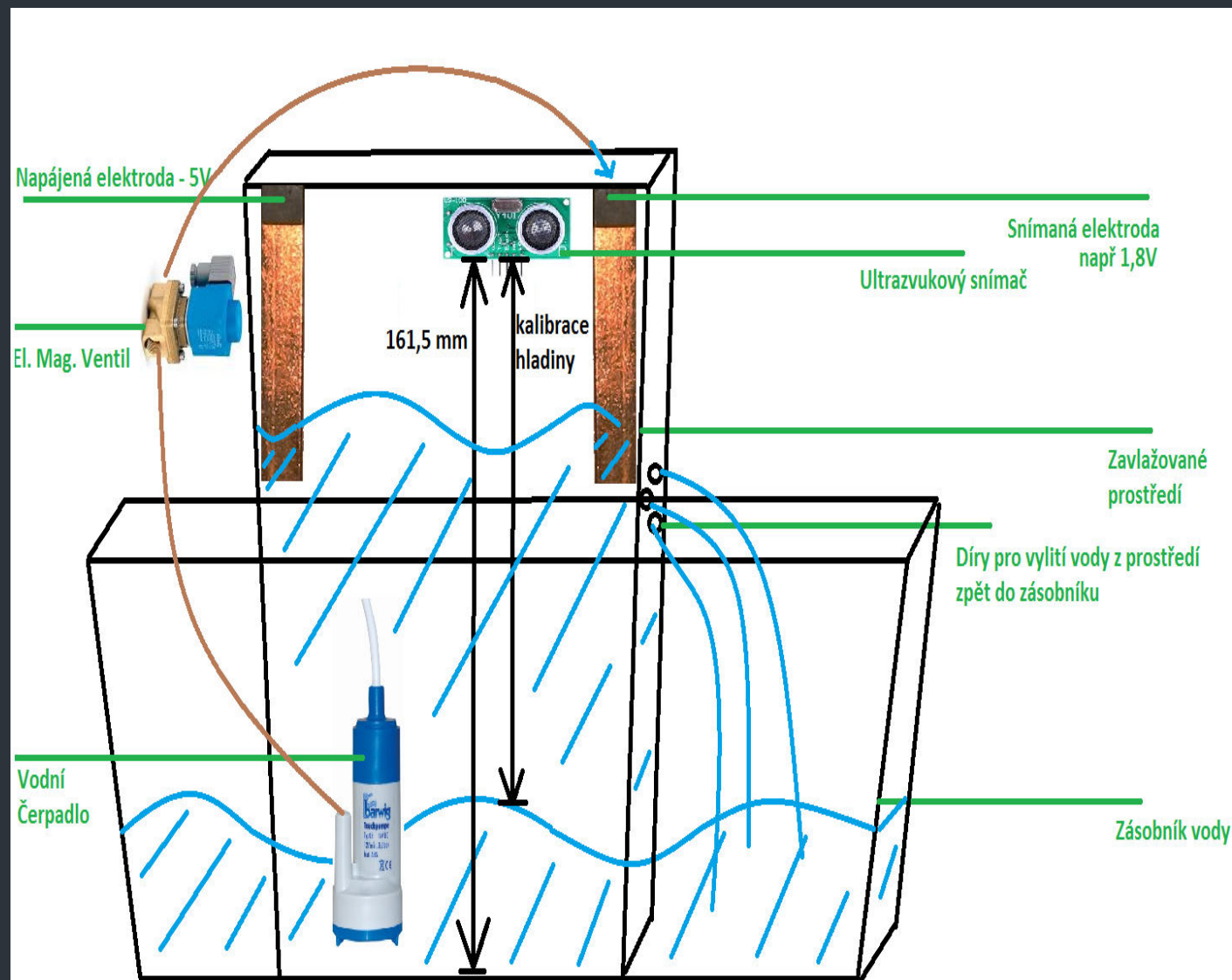
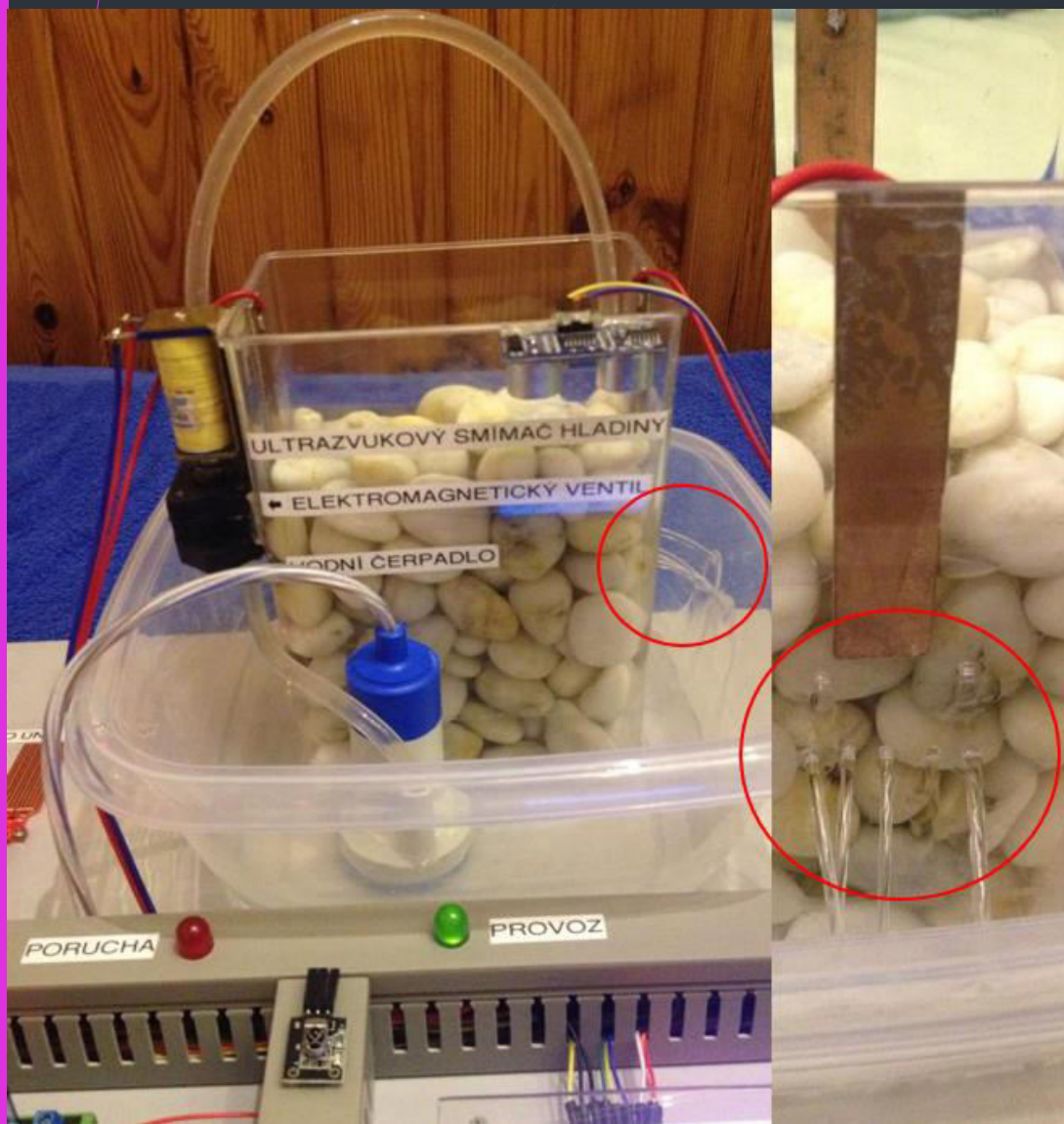
☐ Út

☐ St

## ➤ Nastavení je nutno potvrdit tlačítkem, jinak se neuloží

## ➤ Stránka poté vrací jednotce kód, který se dále zpracovává

# Princip simulace zavlažování



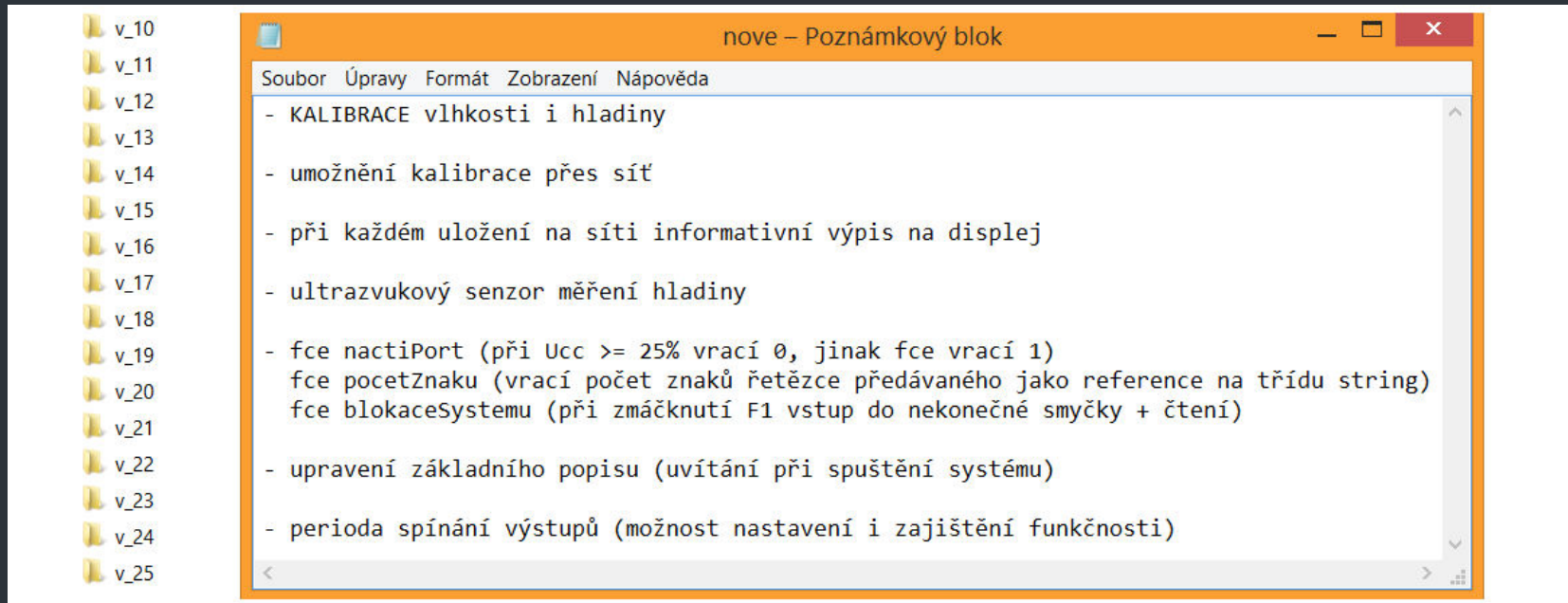


# Technický koutek

- V této kapitole se podíváme na tvorbu softwaru, jak jsou řešeny nejdůležitější procesy a počáteční výběr datových struktur, ale například také na blokové schéma.
- Jsou zde uváděny nejdůležitější úryvky ze zdrojového kódu řídicího programu doplněny o popis a upraveny (zkráceny) pro lepší názornost ukázek.

# Tvorba softwaru

- Program byl vytvořen 16.5.2015 ve vývojovém prostředí Arduino 1.6.6, celkem čítá přes 4700 řádků, zabírá 71 kB (28%) paměti pro program a 7,5 kB (91%) Statické RAM.
- V současné době systém používá 15. verzi 2.5.
- Každá verze obsahuje textový dokument, ve kterém je stručně sepsáno, co nového obsahuje tato verze.





```

class Zapouzdeni_dat
{
private:

    int limitdisplaye = NULL, nastvlhkosti = NULL, nasthladiny = NULL, nasthystereze = NULL, kalibraceVlhkost = NULL, kalibraceHladina = NULL, periodaSpousteni = NULL;

```

```

    typedef struct {
        //struktura pole
        int hodiny1;
        int hodiny2;
        int hodinycelk;
        int minuty1;
        int minuty2;
        int minutycelk;
        int min1;
        int min10;
        int min1celk;
        int min2;
        int min20;
        int min2celk;
    }

```

```

    nastaveniDny;
    nastaveniDny dne[6];
    //strukturové pole

```

```

    int nastavORC[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

```

```

public:
    // METODY / ČLENSKÉ FCE TŘÍDY

```

```

    Zapouzdeni_dat();
    int nastaveniParametru(int limitdisplaye, int nasthladiny, int nastvlhkosti, int nasthystereze);
    int cas(int limitdisplaye);
    int nastavcas(int limitdisplaye);
    int nastavvystupy(int limitdisplaye);
    int vyberRezim();
    int hlaseniPoruchy(bool nastavvystupy1, bool nastavvystupy2, int Binarnivstup1, int i);

```

```

    int zadaniHesla(int adresaPameti, int zadejNeboZmen);
    void nactiVstupy(int analogvstup1, int analogvstup2, int cteniVstupu1, int cteniVstupu2);
    void kalibrace();
    void limit();
    void prectiEEPROM();
    void zapisEEPROM();
    void prace_s_vystupy();
    void zobrazNaDisplayi();
    void stlaceniTlacitkaLoop();
    void ethernet();
    ~Zapouzdeni_dat();
};

```

# Třída Zapouzďření dat

```

//KONSTRUKTOR TŘÍDY

```

```

//fce pro nastavení hladiny,vlhkosti,limitu displaye
//fce pro nastavení Po-Ne (hodiny,minuty,min1,min2)
//fce pro nastavení obvodu reálného času
//fce pro nastavení výstupu (LOW/HIGH)
//fce pro vyber režimu 1 - 2 - 3
//fce pro hlášení poruchy (binární vstup == NULL)

```

```

//slouží pro zadání hesla při startu a při zhasnutí podsvícení displaye
//procedura na načtení hodnot ze vstupů
//nakalibruje dolní a horní mez při přepočtu analogových vstupů
//procedura hlídající vypnutí podsvícení displaye
//procedura, která načte data z EEPROM do proměnných
//procedura, která zapíše data do EEPROM
//procedura, která zajišťuje správný běh výstupů (kdy se sepnou,na jak dlouho)
//procedura má za úlohu vykreslit celé menu, se všemi hodnotami
//procedura, která zajišťuje správnou fci tlačítek
//tato procedura zajišťuje celou sitovou komunikaci
//DESTRUKTOR TŘÍDY

```

# Funkce void loop()

```
void loop()
{

    Zapouzdreni_dat * den = new Zapouzdreni_dat;

    privitani(5000);

    //den->zadaniHesla(9, 1);

    den->nactiVstupy(A14, A12, A15, A13);

    zkontrolujEEPROM();

    for (;;) {

        den->prectiEEPROM();

        den->prace_s_vystupy();

        den->zobrazNaDisplayi();

        problikavani();

        den->limit();

        Binarnivstup1 = nactiPort(A11);

        if ((!Binarnivstup1) && (!pozastavPoruchu))
            den->hlaseniPoruchy(nastavvystupy1, nastavvystupy2, Binarnivstup1, i);

        den->stlaceniTlacitkaLoop();

        den->ethernet();

    }
    delete[] den;
}
```

```
// VYTVOŘENÍ DYNAMICKÉ INSTANCE TŘÍDY NEBOLI OBJEKTU POMOCÍ OPERÁTORU NEW, KVŮLI NEPŘÍMÉNÉ ADRESACI.

// vypíše info o PRG a IP adr, pak počka 3s bez možnosti přeskočení

    // zavolá proceduru, ve které uživatel musí zadat správné heslo, jinak jej nepustí do systému

// funkce která načte hodnoty z analog. a binar. vstupů a uloží je do globálních proměnných

// procedura zapíše nulu na místo,kde je v EEPROM 255

// nekonečná smyčka (stejně jako void loop()), aby se nevytvářely pořád statické proměnné a objekt den

    // načte data z EEPROM

    // kontroluje (zapíná/vypíná) výstupy

    // Zobrazí na Displayi

    // zajišťuje blikání kurzoru a dvojtečky

    // vypnutí podsvícení displaye po uplynutí nastaveného času bez stlačení tlačítka

    // fce vrátí 1 pokud je na portu méně než 25% z 5V a 0 pokud je více nebo rovno 25%.

    // zavolá fci hlášení poruchy, když (binární vstup 1 == NULL) a zároveň pozastav poruchu je neakti

    // Snímá stlačení tlačítka a pro příslušlé tlačítko vykoná příslušné operace

    // zajišťuje síťovou komunikaci

// uvolnění paměti
```

```

void Zapouzdeni_dat::zapisEEPROM()
{
    if ((nasthladiny >= NULL) && (nasthladiny <= 100)) EEPROM.write(0 + rezimEEPROM, nasthladiny);
    if ((nastvlhkosti >= NULL) && (nastvlhkosti <= 100)) EEPROM.write(1 + rezimEEPROM, nastvlhkosti);
    if ((limitdispaye >= NULL) && (limitdispaye <= 60)) EEPROM.write(2 + rezimEEPROM, limitdispaye);
    if ((nedele2 >= NULL) && (nedele2 <= 9)) EEPROM.write(3 + rezimEEPROM, nedele2);
    if ((soucasnyBeh >= NULL) && (soucasnyBeh <= 1)) EEPROM.write(4 + rezimEEPROM, soucasnyBeh);
    if ((nasthystereze >= NULL) && (nasthystereze <= 10)) EEPROM.write(5 + rezimEEPROM, nasthystereze);
    if ((kalibraceVlhkost >= NULL) && (kalibraceVlhkost <= 1023)) EEPROM.write(6 + rezimEEPROM, kalibraceVlhkost);
    if ((kalibraceHladina >= NULL) && (kalibraceHladina <= 1023)) EEPROM.write(7 + rezimEEPROM, kalibraceHladina);
    if ((periodaSpousteni >= NULL) && (periodaSpousteni <= 99)) EEPROM.write(8 + rezimEEPROM, periodaSpousteni);

    int c = 11 + rezimEEPROM;
    for (int a = NULL; a < 7; a++) {
        if ((dny[a].hodiny1 >= NULL) && (dny[a].hodiny1 <= 2)) EEPROM.write(c, dny[a].hodiny1);
        c++;
        if ((dny[a].hodiny2 >= NULL) && (dny[a].hodiny2 <= 9)) EEPROM.write(c, dny[a].hodiny2);
        c++;
        if ((dny[a].hodinycelk >= NULL) && (dny[a].hodinycelk <= 23)) EEPROM.write(c, dny[a].hodinycelk);
        c++;
        if ((dny[a].minuty1 >= NULL) && (dny[a].minuty1 <= 5)) EEPROM.write(c, dny[a].minuty1);
        c++;
        if ((dny[a].minuty2 >= NULL) && (dny[a].minuty2 <= 9)) EEPROM.write(c, dny[a].minuty2);
        c++;
        if ((dny[a].minutycelk >= NULL) && (dny[a].minutycelk <= 59)) EEPROM.write(c, dny[a].minutycelk);
        c++;
        if ((dny[a].min1 >= NULL) && (dny[a].min1 <= 9)) EEPROM.write(c, dny[a].min1);
        c++;
        if ((dny[a].min10 >= NULL) && (dny[a].min10 <= 9)) EEPROM.write(c, dny[a].min10);
        c++;
        if ((dny[a].min1celk >= NULL) && (dny[a].min1celk <= 99)) EEPROM.write(c, dny[a].min1celk);
        c++;
        if ((dny[a].min2 >= NULL) && (dny[a].min2 <= 9)) EEPROM.write(c, dny[a].min2);
        c++;
        if (a != 6) {
            if ((dny[a].min20 >= NULL) && (dny[a].min20 <= 9)) EEPROM.write(c, dny[a].min20);
        }
        c++;
        if ((dny[a].min2celk >= NULL) && (dny[a].min2celk <= 99)) EEPROM.write(c, dny[a].min2celk);
        c++;
    }
}

```

// zápis proměnných do eeprom!

// zápis struktury do eeprom!

# Zápis do paměti EEPROM

```
for (int v = NULL; v <= pocetZnaku(buffer); v++) {
```

```
//-----  
if ((buffer[v] == 'q') && (buffer[v + 1] == '1') && (buffer[v + 2] == '=') && (buffer[v + 3] != '&') && (buffer[v + 4] == '&')) { // hodiny 2  
    dny[vypisDniNaSit].hodiny1 = NULL;  
    dny[vypisDniNaSit].hodiny2 = ((int)buffer[v + 3] - 48);  
    dny[vypisDniNaSit].hodinycelk = ((int)buffer[v + 3] - 48);  
    zapisEEPROM();  
    if (vypisMozny) nastaveniNaEthernetu(2000, 0);  
}
```

```
if ((buffer[v] == 'q') && (buffer[v + 1] == '1') && (buffer[v + 2] == '=') && (buffer[v + 3] != '&') && (buffer[v + 4] != '&') && (buffer[v + 5] == '&')) {  
    dny[vypisDniNaSit].hodiny1 = ((int)buffer[v + 3] - 48);  
    dny[vypisDniNaSit].hodiny2 = ((int)buffer[v + 4] - 48);  
    dny[vypisDniNaSit].hodinycelk = ((10 * ((int)buffer[v + 3] - 48)) + ((int)buffer[v + 4] - 48));  
    zapisEEPROM();  
    if (vypisMozny) nastaveniNaEthernetu(2000, 0);  
}
```

```
//-----  
if ((buffer[v] == 'q') && (buffer[v + 1] == '2') && (buffer[v + 2] == '=') && (buffer[v + 3] != '&') && (buffer[v + 4] == '&')) { // minuty 2  
    dny[vypisDniNaSit].minuty1 = NULL;  
    dny[vypisDniNaSit].minuty2 = ((int)buffer[v + 3] - 48);  
    dny[vypisDniNaSit].minutycelk = ((int)buffer[v + 3] - 48);  
    zapisEEPROM();  
    if (vypisMozny) nastaveniNaEthernetu(2000, 0);  
}
```

```
if ((buffer[v] == 'q') && (buffer[v + 1] == '2') && (buffer[v + 2] == '=') && (buffer[v + 3] != '&'))  
    dny[vypisDniNaSit].minuty1 = ((int)buffer[v + 3] - 48);  
    dny[vypisDniNaSit].minuty2 = ((int)buffer[v + 4] - 48);  
    dny[vypisDniNaSit].minutycelk = ((10 * ((int)buffer[v + 3] - 48)) + ((int)buffer[v + 4] - 48));  
    zapisEEPROM();  
    if (vypisMozny) nastaveniNaEthernetu(2000, 0);  
}
```

## Zpracování dat ze sítě

```
int pocetZnaku(string & retezec) {  
    int pocet = NULL;  
    for (; retezec.c_str()[pocet] != '\0';)  
        pocet++;  
    return pocet;  
}
```















# Stručný výčet všech funkcí systému

- Zadání a změna hesla
- Manuální blokace systému
- Snímání a hlášení poruchy systému
- Měření hladiny vody v nádrži a vlhkosti zavlažovaného prostředí
- Nastavení vlhkosti, hladiny, hystereze, limitu displaye, obvodu reálného času, dní Po-Ne, doby trvání sepnutí 1. a 2. okruhu, současného běhu a periody spouštění výstupů
- Manuální spouštění výstupů
- Výběr režimu (Normal; Sucho; Vlhko)
- Kalibrace výšky hladiny i vlhkosti prostředí
- Měření průtoku vody
- Reset konfigurace do továrního nastavení

# Porovnání této práce s reálným komerčním automatickým zavlažovacím systémem

Funkce	Běžný komerční zavlažovač	Tato práce	Funkce	Běžný komerční zavlažovač	Tato práce
Možnost vzdáleného přístupu			Sedmidenní kalendář s volbou zavlažovacích dní		
Indikace probíhající závlahy			Reset konfigurace systému		
Nastavení délky závlahy			Manuální spouštění sekcí		
Více startů denně			Manuální blokace systému		
Manuální zadání přestávky			Zabezpečení systému		

# Výsledek práce

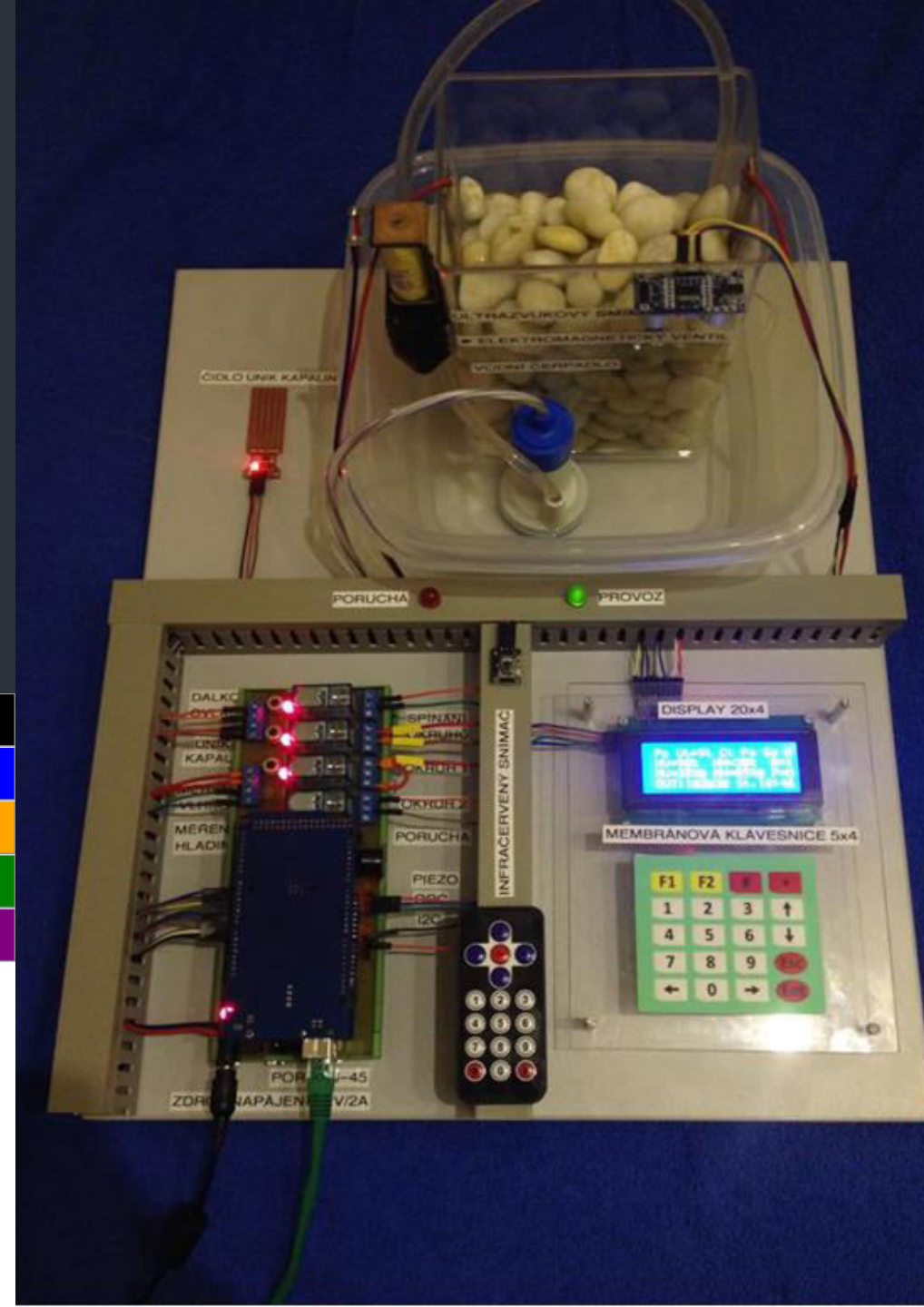
- Plně funkční automatizovaný zavlažovací systém, který je schopen konkurovat obdobným komerčním řešením nabízeným na trhu.

Zavlažovací systém Kočica Filip ME4						
PONDĚLÍ	ÚTERÝ	STŘEDA	ČTVRTEK	PÁTEK	SOBOTA	NEDĚLE
Nastavení parametrů					Režim + Kalibrace	
Vstupy					Porucha rozepnuta	
Výstupy				Aktualní čas		

[Zpět](#)  
Nast. Vlhkost: 90%  
Měřená Vlhkost: 0%  
**PODMÍNKA SPLNĚNA**

Nast. Hladina: 100%  
Měřená Hladina: 100%  
**PODMÍNKA NESPLNĚNA**

Hystereze: 0%  
Limit Displaye: 60m





# Závěr

- Vytvořený model splňuje všechny požadavky stanovené na začátku práce. Je plně funkční a zcela vystihuje veškeré možné situace v zahradách.
- Důraz byl kladený především na názornost celého modelu a zároveň na využití automatizace jako nedílné součásti všech moderních systémů využívaných v dnešní době.
- Celý systém je po krátké instruktáži srozumitelný i pro technicky nezasvěcené osoby.
- Práce pro mě byla výzvou a s jejím výsledkem jsem spokojen.
- Touto tematikou se stále zabývám.

# Použitá literatura a zdroje

- 1. PRATA, Stephen. *Mistrovství v C++*. 4. aktualizované vyd. Brno: COMPUTER PRESS, 2013. ISBN 978-80-251-3828-1.
- 2. Zavlažovací Systémy. In: *azzahrada.cz – Zavlažovací systémy a závlahy* [online]. 2010. vyd. [cit. 2010-12-14]. Dostupné z: <azzahrada.cz>
- 3. ENDRYCH, Václav. *Elektronický systém zavlažování*. Brno, 2011. VUT.



Děkuji za pozornost,  
Kočica Filip.

