iPot

Autor

Vîțoga George-Patrick

Introducere

- Proiectul are ca scop automatizarea unui task comun, dar foarte important, si anume **irigarea plantelor**
- Se doreste implementarea unui ghiveci compact, **capabil sa-si mentina umiditatea** la un nivel optim pentru
  cresterea si dezvoltarea plantelor
- Ghiveciul “iPot” se **auto-iriga** atunci cand este nevoie

Descriere generală

Puncte de interes

> Check the pulse

Ideea din spatele **iPot** este simpla, si urmareaste citirea periodica a umiditatii din ghiveci folosind unul sau mai multi senzori.
[possible optimization]: montarea senzorilor pe mai multe niveluri pe verticala

> Water or not ?

O data ce este cunoscuta umiditatea, datele sunt procesate de Arduino si se ia o decizie. Complexitatea algoritmul din spate poate varia.
[possible optimization]: algoritm cu memorie, deciziile se bazeaza si pe evenimentele din trecut (planta iubitoare de apa vrea sa “bea” mai des ⇒ intelegem cum variaza umiditatea in timp si schimbam strategia de irigare)

> Make it rain

Presupunem ca ne aflam in cazul “planta moare de sete”. Arduino va comanda alimentarea la curent

a pompei, care va avea drept consecinta transferul apei din rezervor in ghiveci.
Comandarea alimentarii se poate face cu ajutorul unui tranzistor mos, dar pentru a izola circuitul de comanda vom folosi un releu.

> Flooding is bad

Exista mai multe strategii de a decide cand ghiveciul a fost irigat suficient, spre exemplu putem efectua **cicluri de mini-irigari si masuratori repetate**, pana cand umiditatea ajunge in parametrii,
sau putem cauta experimental **un interval de timp standard** de irigare.

> Walls have ears too...

O decizie buna in cadrul proiectului este semnalizarea evenimentelor. Putem avertiza sonor utilizatorul cand incepe irigarea sau daca ceva neprevazut apare, spre exemplu, daca nivelul umiditatii din vas este scazut chiar si dupa ce s-a terminat procesul de irigare poate insemna ca nu mai este apa in rezervor.
[possible optimization]: iPot va intra in modul avarie la un eveniment necunoscut Fix Me!
□□□

Schema bloc

Hardware Design

Listă piese

- Arduino (UNO/Nano/Mega)
- Fire, conectori etc.
- Senzor umiditate sol
- Sursă de alimentare
- Pompa de apa
- Buzzer
- Tranzistor Mos
- Rezervor
- Ghiveci

Software Design

Programul care ruleaza pe Arduino poate fi separat in 3 parti.

Logica de redare a muzicii prin buzzer

Ii multumim lui Robson Couto pentru ca ne-a oferit logica pe baza caruia am construit glasul lui iPot.

```
#define NOTE_B0  31
#define NOTE_C1  33
// rest of the notes...

const int melody__irrigationStarted[] = {
  NOTE_C4,4,  NOTE_C4,4,  NOTE_D4,4,  NOTE_E4,4,
  NOTE_E4,-4, NOTE_D4,8,  NOTE_D4,2,
};
const int melody_notes__irrigationStarted = sizeof(melody__irrigationStarted)
/ sizeof(melody__irrigationStarted[0]) / 2;
const int melody_tempo__irrigationStarted = 111;
// rest of the melodies...


void playMelody(
      uint8_t passive_buzzer_pin,           // pin-ul buzzer-ului
      const int melody[],                   // array-ul melodiei format din
mai multe alaturari de nota si duratie
      const int notes,                      // numarul de note din melodie
      const int tempo,                      // viteza melodiei
      bool isBuzzerTriggerLow);             // true - daca buzzer-ul este
lower triggered
```

Se poate observa parametrul **isBuzzerTriggerLow** pe care l-am creat ca o “adaptare” intre codul folosit de pe git si buzzer-ul meu. Aceasta “adaptare” este necesara pentru a putea opera pe un buzzer **low triggered**.
Dupa fiecare nota, trebuie sa oprim buzzer-ul in modul lui de operare corespunzator:

```
if(isBuzzerTriggerLow) digitalWrite(passive_buzzer_pin, HIGH);
```

Citirea senzorilor si calcularea umiditatii

Fiecare senzor este citit de 100 de ori si se face media aritmetica a valorilor citite pentru a gasi un

rezultat mai stabil. Deoarece originea chinezeasca a senzorilor isi face simtita prezenta prin **fluctuatii neasteptate** a fost nevoie sa folosim o solutie romaneasca sa-i **calibram**.

```
sensor_sum += cap(map(map(analogRead(SENSOR_PIN), 1023, 0, 0, 100), 0, 60, 0,
100), 100);
```

Umiditatea din sol este vazuta ca o **medie ponderata** a procentelor stabilite de cei 3 senzori.

- senzorul de sus contribuie 15%
- senzorul din mijloc contribuie 60%
- senzorul de jos contribuie 25%

```
(*soil_volumetric_water) = (15 * (*top_level_humidity) + 60 *
(*mid_level_humidity) + 25 * (*bottom_level_humidity)) / 100;
```

Strategia de irigare

Am ales strategia de irigare care se bazeaza pe **cicluri de mini-irigari si masuratori repetate**.

La inceput se stabileste procentul total de umiditate al solului:

```
  measureSoilHumidity(&top_level_humidity, &mid_level_humidity,
&bottom_level_humidity);
  computeSoilVolumetricWater(&soil_volumetric_water, &top_level_humidity,
&mid_level_humidity, &bottom_level_humidity);
```

Se ia o decizie, bazata pe un prag limita de umiditate:

```
if (soil_volumetric_water < WATER_IRRIGATION_PERCENT_THRESHOLD)
```

Efectuam mai multe cicluri de irigare daca umiditatea este sub pragul limita.

```
while (wateredPot_soil_volumetric_water < WATER_IRRIGATION_PERCENT_THRESHOLD)
```

Un ciclu de irigare este corect daca valoarea umiditatii creste. iPot incearca de 10 ori sa efectueza un ciclu de irigare complet.

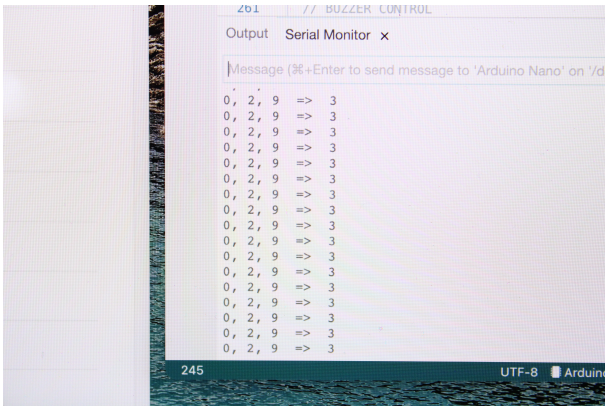
```
for (int try_number = 1; try_number < 10 && !irrigationCycleComplete;
try_number++) {
```

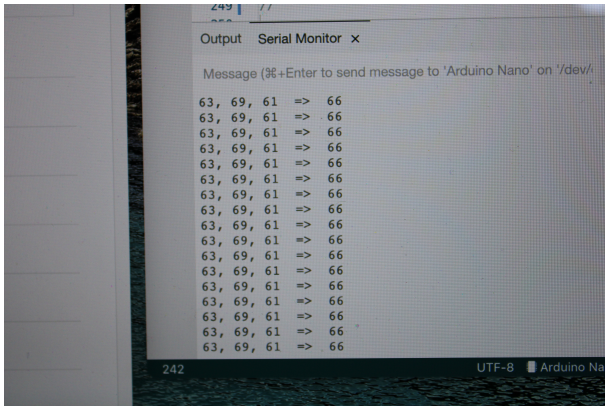
```
        // START WATERING for a little time and after that wait until water
settles
        irrigatePlantForTimePeriod(WATER_IRRIGATION_DURATION_MILLIS);
        delay(WATER_IRRIGATION_TAKE_EFFECT_TIME);

        // measure the new soil volumetric water
        measureSoilHumidity(&top_level_humidity, &mid_level_humidity,
&bottom_level_humidity);
        computeSoilVolumetricWater(&wateredPot_soil_volumetric_water,
&top_level_humidity, &mid_level_humidity, &bottom_level_humidity);

        // check if water reached the pot
        if(soil_volumetric_water < wateredPot_soil_volumetric_water)
irrigationCycleComplete = true;
}

// irrigation cycle failed, human maintenance needed (no water in reservoir)
if(!irrigationCycleComplete) errorMode();
```

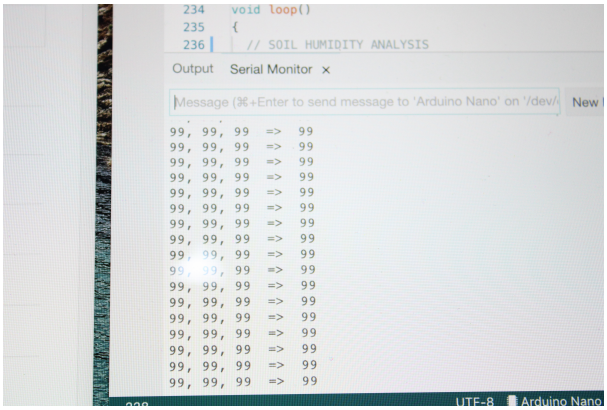
Daca ciclul de irigare nu poate fi finalizat cu succes, errorMode() va trece iPot in modul avarie, asteptand mentenanta umana.
Modul avarie poate reprezenta lipsa apei din rezervor sau alte neajunsuri externe.

Rezultate Obţinute

iPot si-a indeplinit scopul, acesta reusind sa irige atunci cand planta din vas are nevoie de apa.
Mai jos sunt cateva imagini cu datele de la senzori si umiditatea din vas pe tot parcursul irigarii unei rosii abia plantate intr-un sol nou si uscat.







Prezentare: youtube

Concluzii

iPot este un proiect bun si interesant pentru a invata PM. Totodata este si **foarte edificator**, deoarece iti arata toatalitatea lucrurile care pot sa mearga rau intr-un task simplu, lucruri de care nu s-a tinut cont.

- Senzorii de umiditate ruginesc
- Senzorii de umiditate sunt imprecisi
- Firele scufundate in apa trebuie izolate **perfect**
- Arduino este foarte util, mai ales cand nu alimentezi motoare cu el...
- Buzzer-ul chinezesc este low triggered, deci tone() si noTone() nu mai merg cum te astepti

Acestea au fost doar cateva din lucrurile minunate descoperite pe parcursul proiectului.
Pentru efectuarea prototipului **s-au investit**: 100 lei + 25 ore de munca

> **Mentiune**: iPot nu este fiabil (senzorii vor rugini) si nici nu merge folosit cu orice planta.

Download

archive.zip

Jurnal

- 25 Aprilie - Alegere tema proiect
- 27 Aprilie - Acceptul laborantului + inlocuire releu cu tranzistor mos
- 17 Mai - Citire senzori si control pompa de apa reusite
- 20 Mai - Codul finalizat si testat cu succes pe sol fara planta
- 22 Mai - Componentele hardware asezate in pozitia lor finala
- 23 Mai - Finalizare documentatie ocw + prezentare youtube

Bibliografie/Resurse

iPot
github - souce code and more details
youtube - prezentare si testare
Robson Couto: arduino-songs
Export as PDF