INCLUSIVE ROBOTICS · 2024-2· Final Project Report

Intelligent Hydraulic Monitoring and Control System

for Automatic Irrigation with Arduino

Authors

- Rafael Sampaio e Silva and Tarsila Amado Alves de Brito

Resumo

This project aims to develop a self-watering pot system to simplify plant care for individuals with busy routines and elderly people with Alzheimer's, promoting well-being through contact with nature. The system utilizes a robot equipped with the FC-28 Soil Moisture Sensor to monitor soil moisture levels and a 3-6V RS-385 Mini Water Pump for automatic irrigation. The innovation lies in the ability to configure the irrigation frequency, allowing each plant to receive personalized care, even in the absence of constant supervision, optimizing water usage and ensuring plant health.

Palavras-chave: Automation, Monitoring, Self-sustaining, Self-watering.

1 Introduction

The presence of plants in indoor environments is associated with physical and psychological benefits, especially for elderly individuals with Alzheimer's, as contact with nature helps in sensory and emotional stimulation. However, caring for plants can be a challenge for individuals with mobility, memory difficulties, or limited time for constant watering. Aiming to provide a practical solution, this project proposes the development of a self-watering pot that monitors soil moisture levels and performs irrigation autonomously and in a personalized manner. The system allows adjusting the irrigation frequency, adapting to the specific needs of different plant species and providing ease and reliability in care.

2 Experiment Description

For the implementation of the self-watering pot, an FC-28 Soil Moisture Sensor was used, responsible for measuring the soil moisture level. The sensor reading is processed by a controller, which, according to the configuration set by the user, activates a 3-6V RS-385 Mini Water Pump for irrigation. The system allows configuring specific irrigation frequencies for different types of plants, ensuring that irrigation occurs only at the appropriate time, even when the sensor detects low moisture. In this way, the user can choose the ideal watering intervals for each plant, ensuring its health and optimizing water consumption.

3 Materials and Algorithm Versions

3.1 System Version 1

The first version of the self-watering pot system was developed in an online simulation (Thinkerkad) with the initial goal of creating basic automatic irrigation. This version used the FC-28 Soil Moisture Sensor to measure soil moisture, however, the irrigation frequency was not configurable, and the water pump had not yet been added to the project. Although it provided a functional base, this version was very incomplete, leading to the need for improvements.

3.2 System Version 2

In the second version, the system was updated to include irrigation frequency control, allowing the user to choose between three options: Low, Medium, and High Frequency. For each of these frequencies, the irrigation time (i.e., the amount of water used to water the plant) was defined, addressing our goal of caring for different types of plants. Despite this, the 3-6V RS-385 Mini Water Pump had not yet been incorporated into the system due to the future need for infrastructure adjustments in the software and testing of the electronic components under real conditions (*Results – Experiment 3*). Finally, in the online simulation of this version, a temporary servo motor was added to simulate the opening of a valve or activation of the pump that will be used in the future.

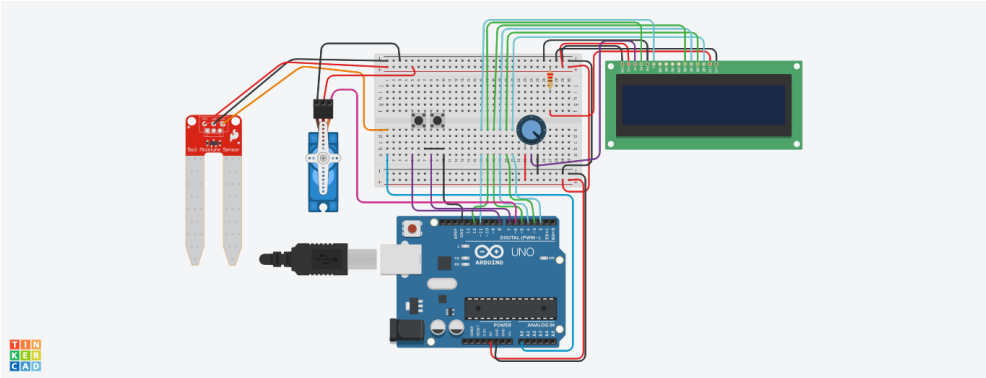

Figura 1: Circuit Structured in Tinkercad

3.3 System Version 3

In the third version, the system was put into practice with experiments (*Results – Experiment 2*) conducted in the lab to test the components under real conditions. After the tests, it was necessary to modify and improve the algorithm, including the addition of a timer and changes to the frequency feature, which was updated to High, Daily, and Every 2 Days. Additionally, the 3-6V RS-385 Mini Water Pump was finally incorporated into the system, automatically activated by the moisture sensor when the moisture level falls below 40%, or if the chosen frequency meets the required conditions.

3.4 Algorithms

Listing 1: Code Version 1

```
#include <LiquidCrystal.h>

#define umidadeAnalog A0
#define pushbuttonOptions 7
#define pushbuttonDone 8

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int humidityValue;
int nivelDeIrrigacao = 1;
bool configurandoNivel = false;
bool seJaRegou = false;
bool sePodeRegar = false;
int lastButtonStateOptions = HIGH;
int lastButtonStateDone = HIGH;

void MenuLCD(){
  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.print("Humidade: ");
```

```
22    lcd.print(humidityValue);
23
24    lcd.setCursor(0, 1);
25
26    // Exibir a frequencia de irrigacao selecionada
27    if(nivelDeIrrigacao == 1){
28      lcd.print("Freq: BAIXA");
29    }else
30    if(nivelDeIrrigacao == 2){
31      lcd.print("Freq: MEDIA");
32    }else
33    if(nivelDeIrrigacao == 3){
34      lcd.print("Freq: ALTA");
35    }
36  }
37
38  void ConfigurandoMenu() {
39    lcd.clear();
40    lcd.setCursor(0, 0);
41    lcd.print("Usar Frequen:");
42
43    lcd.setCursor(0, 1);
44
45    // Mostrar a frequencia atual
46    if(nivelDeIrrigacao == 1){
47      lcd.print("--->BAIXA");
48
49    }else
50    if(nivelDeIrrigacao == 2){
51      lcd.print("--->MEDIA");
52    } else
53    if(nivelDeIrrigacao == 3){
54      lcd.print("--->ALTA");
55    }
56  }
57
58  void abrirValvula(){
59  }
60  void fecharValvula(){
61  }
62
63  void setup(){
64    pinMode(umidadeAnalog, INPUT);
65    pinMode(pushbuttonOptions, INPUT_PULLUP);
66    pinMode(pushbuttonDone, INPUT_PULLUP);
67
68    Serial.begin(9600);
69    lcd.begin(16, 2);
70  }
71
72  void loop() {
73    fecharValvula();
74
75    humidityValue = analogRead(umidadeAnalog);
76    humidityValue = map(humidityValue, 1023, 315, 0, 100);
77    Serial.print("Humidity: ");
78    Serial.print(humidityValue);
79
80    int buttonStateOptions = digitalRead(pushbuttonOptions);
81    if(buttonStateOptions == LOW && lastButtonStateOptions == HIGH){
82      configurandoNivel = !configurandoNivel;
```

```
83      delay(200);
84    }
85    lastButtonStateOptions = buttonStateOptions;
86
87    if(configurandoNivel){
88      ConfigurandoMenu();
89
90      int buttonStateDone = digitalRead(pushbuttonDone);
91
92      if(buttonStateDone == LOW && lastButtonStateDone == HIGH){
93        nivelDeIrrigacao++;
94        if(nivelDeIrrigacao > 3){
95          nivelDeIrrigacao = 1;// Resetar para BAIXA
96        }
97        delay(200);// Debounce
98      }
99      lastButtonStateDone = buttonStateDone;
100   }
101   else{
102     MenuLCD();
103   }
104   delay(100);
105
106   humidityValue = analogRead(umidadeAnalog);
107   humidityValue = map(humidityValue, 1023, 315, 0, 100);
108   if(humidityValue <= 40){
109     abrirValvula();
110     delay(600);
111     fecharValvula();
112     delay(2000);
113   }else if(humidityValue <= 70){ // Ja esta Humido
114     fecharValvula();
115   }
116 }
```

Listing 2: Code Version 3 – Final

```
1  #include <LiquidCrystal.h>
2
3  #define umidadeAnalog A0
4  #define pushbuttonOptions 7
5  #define pushbuttonDone 8
6  #define relePin 9
7
8  LiquidCrystal lcd(2, 3, 4, 5, 11, 12);
9
10 int humidityValue;
11 int nivelDeIrrigacao = 1;
12 bool configurandoNivel = false;
13 unsigned long ultimoTempoDeIrrigacao = 0;
14 unsigned long intervaloDeIrrigacao = 0;
15
16 // Definir os intervalos em milissegundos
17 #define INTERVALO_DIARIO 86400000
18 #define INTERVALO_2DIAS 172800000
19
20 void MenuLCD(){
21   lcd.clear();
22   lcd.setCursor(0, 0);
23   lcd.print("Humidade: ");
24   lcd.print(humidityValue);
```

```
25    lcd.setCursor(0, 1);
26
27    if(nivelDeIrrigacao == 1){
28      lcd.print("Frequencia: ALTA");
29    }else
30    if(nivelDeIrrigacao == 2){
31      lcd.print("Frequencia:DIARIA");
32    }else
33    if(nivelDeIrrigacao == 3){
34      lcd.print("Frequencia:2DIAS");
35    }
36  }
37  void ConfigurandoMenu(){
38    lcd.clear();
39    lcd.setCursor(0, 0);
40    lcd.print("Usar Frequencia:");
41    lcd.setCursor(0, 1);
42
43    if (nivelDeIrrigacao == 1){
44      lcd.print("--->ALTA");
45    }else
46    if(nivelDeIrrigacao == 2){
47      lcd.print("--->DIARIA");
48    }else
49    if(nivelDeIrrigacao == 3){
50      lcd.print("--->2 DIAS");
51    }
52  }
53  void abrirValvula(){
54    digitalWrite(relePin, HIGH);
55  }
56  void fecharValvula(){
57    digitalWrite(relePin, LOW);
58  }
59
60  void abrirBombaRapido() {
61    digitalWrite(relePin, LOW);
62    unsigned long tempoBomba = millis();
63    while (millis() - tempoBomba < 5000) {
64
65      int buttonStateOptions = digitalRead(pushbuttonOptions);
66      int buttonStateDone = digitalRead(pushbuttonDone);
67
68      if (buttonStateOptions == LOW || buttonStateDone == LOW) {
69
70        configurandoNivel = true;
71        fecharValvula();
72        return;
73      }
74      delay(100);
75    }
76    digitalWrite(relePin, HIGH);
77    delay(1000);
78  }
79
80  void loop() {
81    humidityValue = analogRead(umidadeAnalog);
82    humidityValue = map(humidityValue, 1023, 315, 0, 100);
83
84    // Leitura e alternancia do estado dos botoes
85    int buttonStateOptions = digitalRead(pushbuttonOptions);
```

```
  static int lastButtonStateOptions = HIGH;
  if (buttonStateOptions == LOW && lastButtonStateOptions == HIGH) {
    configurandoNivel = !configurandoNivel;
    delay(200);
  }
  lastButtonStateOptions = buttonStateOptions;

  // Modo configuracao
  if (configurandoNivel) {
    ConfigurandoMenu();

    int buttonStateDone = digitalRead(pushbuttonDone);
    static int lastButtonStateDone = HIGH;
    if (buttonStateDone == LOW && lastButtonStateDone == HIGH) {
      nivelDeIrrigacao++;
      if (nivelDeIrrigacao > 3) {
        nivelDeIrrigacao = 1;
      }
      delay(200); // Debounce
    }
    lastButtonStateDone = buttonStateDone;

  } else {
    // Exibe menu LCD no modo padrao
    MenuLCD();

    unsigned long tempoAtual = millis();

    // Logica de irrigacao ======================================
    // Modo ALTA
    if (nivelDeIrrigacao == 1) {
      if (humidityValue < 40) {
        abrirBombaRapido();
      }
     // Modo DIARIA
    } else if (nivelDeIrrigacao == 2) {
      intervaloDeIrrigacao = INTERVALO_DIARIO;
      if ((tempoAtual - ultimoTempoDeIrrigacao >= intervaloDeIrrigacao)
      && humidityValue < 40) {
        abrirBombaRapido();
        ultimoTempoDeIrrigacao = tempoAtual;
      }
    } else if (nivelDeIrrigacao == 3) { // Modo CADA 2 DIAS
      intervaloDeIrrigacao = INTERVALO_2DIAS;
      if ((tempoAtual - ultimoTempoDeIrrigacao >= intervaloDeIrrigacao)
      && humidityValue < 40) {
        abrirBombaRapido();
        ultimoTempoDeIrrigacao = tempoAtual;
      }
    }
  }

  delay(100);
}
```

3.5 Methodology, Structure, and Functioning

During the construction of the robot prototype, the V-Model Systems Engineering Methodology was adopted. This model is suitable for the project because it is commonly used in hardware and software projects. It organizes the work into sequential stages but allows iterations and testing before progressing. Thus, the work

was divided into three stages:

- Definition and Planning (Phase 1): Design the system in Tinkercad, including simulation.
- Prototyping and Preliminary Testing (Phase 2): Initial circuit construction, algorithm development, and error identification.
- Integration and Final Testing (Phase 3): Final prototype construction, where all components are integrated.

As a prototype, the 'irrigation station' was built. The robot is approximately 40 cm tall, 30 cm wide, and 30 cm long. It was designed with a functional division into two main sections: the lower half houses the water reservoir, responsible for storing the liquid used for plant irrigation, from which a small pipe (25 mm) extends, connected to the plant pot. The upper half contains the electronic components of the system, including the Arduino, control buttons for the LCD display, and, at the top, the moisture sensor, which is directly connected to the soil in the plant pot.

Bill of Materials:

- Arduino Uno
- PCB Protoboard
- 16x2 LCD Display
- Jumper Wires
- Mini Water Pump (120L, 3V to 6V)
- Diffuser LM393
- Soil Moisture Sensor
- Relay Module MPA-S-105-C
- 10k Potentiometer
- 2x 220 Ohm Resistors
- 2x 4-Pin Microswitch Push Buttons
- 9V Battery Connector
- 9V Battery
- 4-AAA Battery Holder (1.5V each)
- 4 AAA Batteries (1.5V each)
- 2x Plastic Cake Containers
- 25mm Aquarium Tube
- Electrical Tape

3.6 Flowchart

[Figure 2, illustrated on the next page]

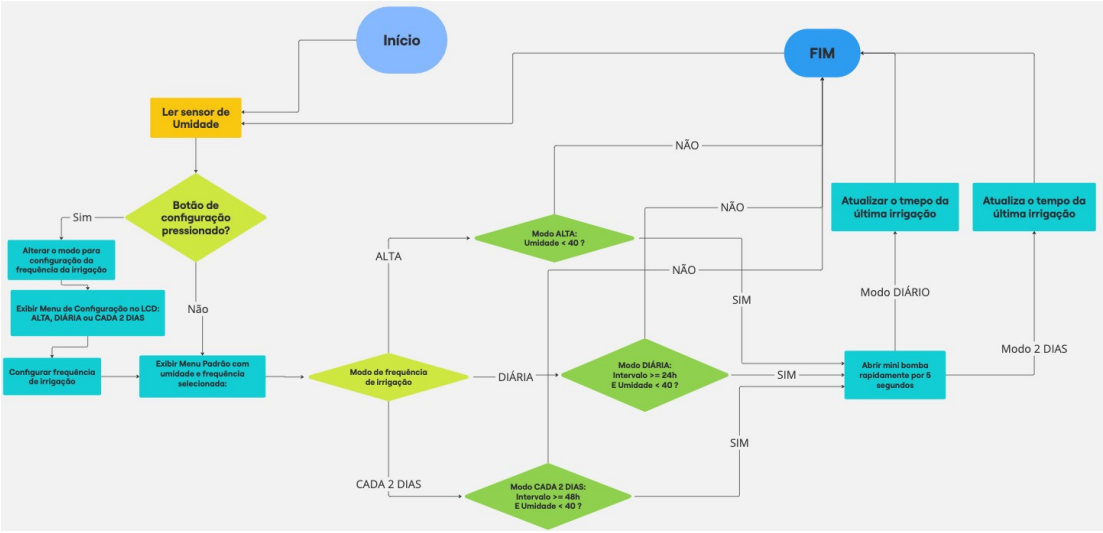

Figura 2: Flowchart of the final code

4 Results

4.1 Results – Experiment (system in version 3)

The experiment was designed to obtain the necessary soil moisture values in different states. During the classroom experiment, we had 4 containers, each with soil in different irrigation states: The first container was never watered, the second was watered within the last 48 hours, the third was watered within the last 24 hours, and the fourth was watered minutes before the sensor measurement. In this way, with the collected data, the need for algorithm adjustments was observed.

After the implementation of the code and the assembly of the irrigation system with the specified components, the automatic irrigation system successfully functioned. The FC-28 soil moisture sensor detects the soil moisture level and activates the 12V RS-385 Mini Water Pump according to the pre-established conditions, allowing personalized irrigation for different types of plants.

The moisture sensor operates according to the following cases:

- Case 1: When the moisture level is less than or equal to 40
- Case 2: When the moisture level is greater than 40

These values were determined to ensure that the plant receives water only when necessary, preventing over-irrigation. On the control screen, the user can configure the irrigation frequency with three options:

- High Frequency: irrigation occurs several times a day.
- Daily: irrigation occurs once per day (24 hours).
- Every 2 Days: irrigation occurs every two days (72 hours).

The tests demonstrated that the system responds efficiently to the frequency settings chosen by the user, providing flexibility in plant care and promoting optimized water usage.

5 Concluding Remarks

The development of this self-watering planter achieved the goal of simplifying plant care, offering a practical and efficient solution for people with busy routines and for the elderly, especially those with Alzheimer's, who can benefit from contact with nature. The system demonstrated the ability to monitor soil moisture and perform automated, personalized irrigation, with frequency settings adjustable by the user.

The tests showed that the system responds well to variations in moisture and frequency settings, promoting optimal plant hydration while avoiding both excessive and insufficient irrigation. With the adjustment of three frequency levels (high, daily, and every two days), the system adapts to the specific needs of different plant types, making care easier and more accessible.

This project represents an important step in the application of simple and accessible technologies to improve quality of life and promote well-being through nature. In future projects, the system could be expanded with additional sensors and remote connectivity, enabling monitoring and control via mobile devices, making the solution even more robust and user-friendly."