**Embedded Irrigation**

Embedded C project using AVR-based MCU to estimate irrigation times and durations based on environmental factors.

# User Story

Keeping a plant alive can be an overwhelming process for beginners and children since there are many factors to consider during the irrigation process. EmbeddedIrrigation helps you provide healthy irrigation for your plants through calculating the number of irrigations per day and the duration of each irrigation based on three factors:

* The amount of water the plant needs: user-adjusted through potentiometer knob and 7-segments display.
* The current temperature: automatically measured through an LM35 analog sensor.
* The current illumination level: automatically measured through an LDR.

# Program Flow

The program has two modes:

* Normal mode

In the normal mode, the temperature and brightness sensors are actively taking readings and, accordingly, printing to the Serial monitor and adjusting the status of the two LEDs. Iff temperature is above 46, the temperature LED will light up. Iff the brightness is below 850, the other LED will light up.

* Irrigation mode

The irrigation mode can be initialized by pressing the button. In the irrigation mode, the user is prompted with a welcome message for starting a new plant and is asked to turn the knob (potentiometer) to indicate the level of water the plant needs. The user can read the amount of water currently selected on the 7-segments display. Once the user's selection is final, they can press the button again to indicate so. The program calculates the number and durations of irrigations needed per day.<p>

Then, the number of irrigations needed per day, along with the current temperature, brightness, and other instruction, will show up on the serial monitor. The alert sequence will go off at the same time to indicate that the user must start the irrigation process now. To stop the alert and start the timer, the user must hit the button. After the calculated duration, the alert sequence will go off again to have the user stop irrigating, or else the plant will die. Once the user stops irrigating, they should hit the button again to stop the alert and exit irrigation mode.

# Testcases

2 types of testing were conducted at different stages of the hardware implementation of the project:

* Whitebox testing: Each software and hardware component was tested separately to ensure that the end-system would have to underlying bugs.
  + The software components being tested were the following:
    - **Dio library:** through trying all the Arduino uno pins as both input and output using a dummy led and pushbutton to write and read, respectively, using the library functions with different ways to call it. For example, I once called SetPinStatus('D', 7, 'I) and another time I tested SetPinStatus('D', 7, 0).
    - **Uart library**: through printing and reading different characters and character arrays from and to the Serial monitor. I tried entering multiple values for characters that also include integers to notice the respective ascii behavior.
    - **Adc library:** by trying to read different sensors on different channels at the same time. I found out that the accuracy of the ADC is better when only one sensor is connected (or the mux doesn't have to change channels very frequently). Therefore, I edited the library so as to close the ADC and re-initialize before each conversion. This enhanced the accuracy of the conversion, compromising speed.
  + The hardware component testing included the following:
    - **Pushbutton system:** the pull-up circuit was tested first using the built-in Arduino functions. I then tested the whole system by printing to the Serial monitor using the implemented libraries.
    - **Led system:** different values for resistance were used to make sure the Arduino peripherals are not damages while preserving sufficient brightness level for the LEDs.
    - Buzzer system
    - **7-segments display system:** a demo 0 to 9 counter was implemented using Arduino built-ins which was then converted to purely C using the Dio library.
    - **LDR system:** tested by changing brightness to upper and lower ones and seeing the respective change on the LDR voltage reading.
    - **The LM35 system:** First tested by changing the temperature and noticing the amount of change to the voltage read. Also tested by not changing the temperature and checking if the readings will stay roughly the same.
    - **Potentiometer system:** by turning the knob of the potentiometer and keeping track of the printed value to note the range of the output voltage.
* Blackbox testing: The system was tested as a whole to check if the expected behavior was achieved.
  + **Normal mode:** by not pressing the button and watching the readings of the LDR and LM35 get printed on the Serial monitor. Also, by covering the LDR and LM35, simultaneously and separately, and watching the LEDs light up when the value is outside of the threshold boundaries.
  + **Irrigation mode:** by trying different combinations of light and heat to see if the duration of the irrigation changes. Also, by truning the knob of the potentiometer to see the respective mapped value on the 7-segments. Finally, by trying to press the button or turn the knob when the program is not expecting the user to interact with the system and watching the no unexpected behavior occurs.
  + **Normal/Irrigation mode interaction:** by entering the irrigation mode, fully testing it till it exits, then re-entering it again multiple times, while testing the normal mode in between periods of irrigation mode to make sure they're integrated together.

# BoM

Total Cost: 212 EGP

|  |  |  |
| --- | --- | --- |
| Component | Price in EGP per unit | Total price in EGP |
| Arduino Uno | 125 | 125 |
| Breadboard | 20 | 20 |
| Jumper wires | 0.75 | 15 |
| Resistors | 0.25 | 1 |
| Pushbutton | 1 | 1 |
| LEDs | 0.5 | 1 |
| Potentiometer | 4 | 4 |
| LDR | 10 | 10 |
| LM35 | 25 | 25 |
| 7-segments display | 5 | 5 |
| Buzzer | 5 | 5 |

# Video Demo

The video demo can be found at: <https://drive.google.com/file/d/1ATuG1tQZMa_KHwsERBIh5QhZrqEs1mak/view?usp=sharing>

# Diagram Description automatically generated with medium confidenceSchematic diagram

# Software Design Architecture

Diagram

Description automatically generated